Small Reasoning Models are Instruction Followers in Function Calling

Yalda Taheri
Department of Engineering
Islamic Azad University
y.jaliltahei@iau.ir

Mohammad Hassan Heydari
Department of Computer Engineering
University of Isfahan
m.heydari@mehr.ui.ac.ir

Erfan Naaman
Department of Computer Engineering
University of Isfahan
erfannamaan@mehr.ui.ac.ir

Afsaneh Fatemi
Department of Computer Engineering
University of Isfahan
a_fatemi@eng.ui.ac.ir

Abstract

Function calling represents the core capability of agentic large language models (LLMs). Existing research has focused on enhancing LLMs' function-calling accuracy through fine-tuning, reinforcement learning (RL), and multi-agent frameworks, particularly for native function-calling LLMs. This work demonstrates that LLMs achieve superior accuracy in function calling in instruction-following contexts (i.e., standard user-assistant interactions) rather than a tool calling context. We introduce Instruction-Followed Function Calling (IFFC), a novel framework that decouples function-calling logic from the primary LLM and delegates it to a dedicated smaller model operating within the instruction-following paradigm. Our method consistently outperforms both native function calling (NFC) and prompt-based function calling (PFC) baselines, with particularly strong gains on reasoning-oriented LLMs. Furthermore, we demonstrate that IFFC maintains robust performance under aggressive quantization, enabling efficient on-device deployment without significant accuracy degradation. This work establishes a new paradigm for reliable, resource-efficient function calling in edge-computing scenarios.

1 Introduction

The evolution of artificial intelligence has transitioned from static text generation to autonomous, active problem-solving, a paradigm known as "Agentic AI" (Patil et al., 2025). At the core of this transition is function calling, which enables models to interact with external environments by selecting tools, generating arguments, and executing actions (Kavathekar et al., 2025). While massive proprietary models initially dominated this space, there is a growing shift toward specialized, task-specific agents that are often more suitable for structured and repetitive workflows than generalist architectures (Belcak et al., 2025; Zeng et al., 2025). Simultaneously, Small Language Models (SLMs) ranging from 0.5 to 15 billion parameters have emerged as a practical alternative to massive models, which often suffer from high computational demands, latency, and privacy risks (Samoylenko; Xu et al., 2024). When specialized, these compact models can rival their larger counterparts in domain-specific applications, such as fault diagnosis and code generation, while remaining accessible to edge devices and common hardware (Kumar et al., 2025; Nath et al., 2025; Sinha et al., 2025). This makes the intersection of SLMs and localized agentic workflows a highly promising area of deployment. However, enabling robust function calling within SLMs presents distinct challenges, as these smaller models frequently struggle with rigid syntactic constraints, complex JSON schemas, and multi-step reasoning (Kavathekar et al., 2025; Sharma and Mehta, 2025). Although techniques such as targeted fine-tuning, reinforcement learning, and advanced prompt engineering have been proposed to mitigate these limitations (Jhandi et al., 2025; Paprunia et al., 2025; Han et al., 2025), many of these approaches continue to force SLMs into native function-calling paradigms designed for larger models. Consequently, this often leads to suboptimal format adherence and fragile execution logic in constrained environments.

In this paper, we challenge the prevailing reliance on native function calling (NFC) for agentic tasks. We posit that the architectural strengths of language models, particularly those optimized for reasoning, lie in following natural language instructions rather than manipulating rigid tool definitions (Johnson et al., 2025). We introduce Instruction-Followed Function Calling (IFFC), a framework that re-imagines the tool-use process by decoupling the function calling task from the main question-answering (QA) model and assign it to another model (in our case, an SLM) which separates their contexts entirely (Roth et al., 2025; Jeon

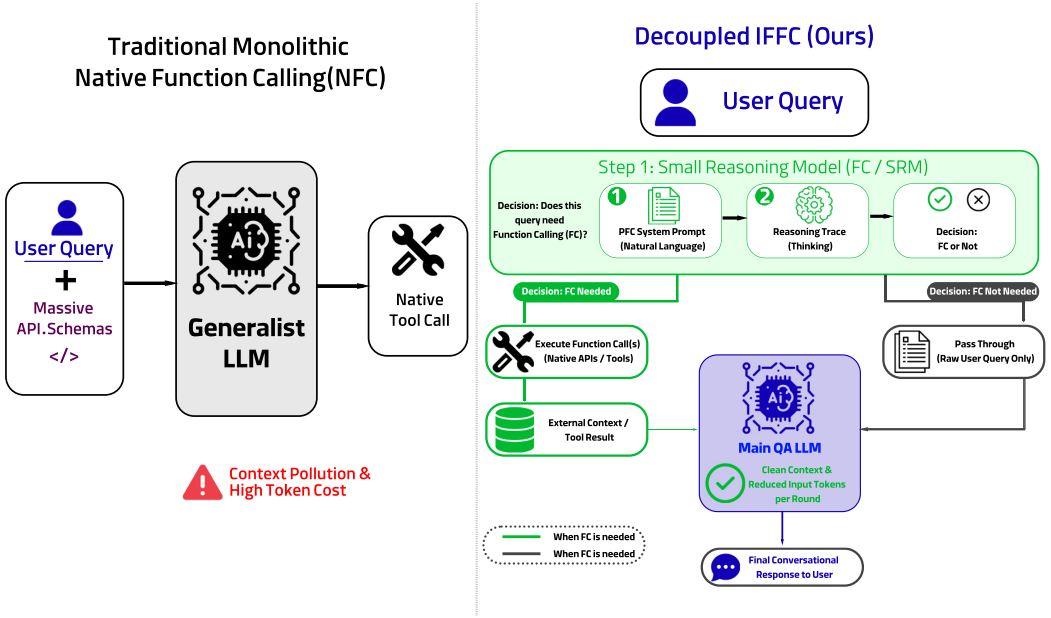

Figure 1: Overview of Instruction-Followed Function Calling (IFFC) framework against traditional function calling approach.

et al., 2025). By delegating the function-calling mechanism to a dedicated, instruction-following paradigm with Reason-Action (ReAct) mechanism, we bypass the limitations inherent in native tool support. Our contributions are as follows:

- We demonstrate that, contrary to common practice, models achieve superior function-calling accuracy in instruction-following contexts (a.k.a. regular user-assistant prompting context) compared to native tool-calling contexts.
- We propose the IFFC framework, which first uses PFC as its tool calling method and then enables smaller reasoning models (SRMs) to execute complex tool usage with higher reliability than native baselines, while being completely separated from the main QA LLM. We show that this context separation allows the SRM to be less context polluted by the outputs of the QA LLM. This hypothesis, which assumes that reasoning models perform better than non-reasoning models, is validated throughout our experiments
- We validate the efficiency of our approach by showing that IFFC maintains robust performance even under aggressive quantization. This establishes a viable path for deploying accurate, privacy-preserving agentic capabilities on resource-constrained edge devices.

2 Related Works

Function calling has transitioned language models from passive text generators to active agents capable of automating complex tasks (Kavathekar et al., 2025). While early agentic frameworks relied on massive, general-purpose models, recent research emphasizes that small language models (SLMs) under 7 billion parameters offer a more stable, cost-effective, and private alternative for structured workflows (Sharma and Mehta, 2025; Samoylenko). When properly specialized, these compact architectures can rival larger systems on targeted reasoning tasks, bypassing the significant computational and financial overhead associated with proprietary models (Sinha et al., 2025; Xu et al., 2024). Deploying SLMs for function calling introduces unique architectural opportunities, such as guided decoding, type-safe registries, and decoupled structures that separate planning from execution to minimize formatting errors (Sharma and Mehta, 2025; Roth et al., 2025). These optimizations enable efficient local deployment on resource-constrained edge devices, as demonstrated by frameworks like TinyAgent (Erdogan et al., 2024). Furthermore, techniques like quantization and domain-specific adaptation allow SLMs to excel in low-latency and privacy-sensitive applications, including automotive control, smart home assistance, and specialized domain tasks (Khiabani et al., 2025; Huang et al., 2026; Jia et al., 2025;

Nath et al., 2025). To narrow the performance gap between SLMs and larger models, researchers have investigated both training-based and training-free optimization strategies. Training-based methods leverage reinforcement learning, group relative policy optimization (GRPO), domain-specific fine-tuning, and planning distillation from larger teacher agents to reinforce structured tool-use behavior (Qian et al., 2025; Paprunia et al., 2025; Jhandi et al., 2025; Qiu et al., 2025). Alternatively, training-free approaches utilize natural language tool interfaces, advanced prompt engineering, reasoning blueprints, and iterative trial-and-error frameworks to improve tool selection and execution without additional training overhead (Johnson et al., 2025; He, 2024; Han et al., 2025; Qu et al., 2024).

3 Methodology

We introduce a two-stage paradigm: first, decoupling the tool-selection logic from the primary generation model, and second, re-framing the function-calling task as a standard instruction-following interaction.

3.1 Decoupling Function Calling from the Main LLM

Traditional agentic workflows often utilize a single monolithic LLM to handle both the reasoning required for tool selection and the final synthesis of the response. This approach frequently leads to context pollution, where the presence of complex API schemas in the prompt interferes with the model's conversational performance or reasoning depth (Roth et al., 2025; Patil et al., 2025).

Drawing inspiration from the success of multi-agent systems in isolating specific sub-tasks (Zeng et al., 2025), we propose a decoupled agentic framework as illustrated in Figure 1 and Algorithm 1. We introduce a dedicated **SRM**, typically ranging from 0.5B to 15B parameters, to serve as the primary routing and tool-execution layer. When an *Incoming Query* is received, the SRM evaluates the user's intent to determine if external context or tool execution is required to provide an accurate answer.

By delegating this logic to a smaller, faster, and more cost-effective model, we achieve several advantages:

- **Context Isolation:** The main LLM's context remains focused on the user interaction and the final response generation, while the SRM handles the "heavy lifting" of tool-definition parsing (Belcak et al., 2025).
- **Efficiency:** The SRM can be aggressively quantized and deployed on the edge, significantly reducing latency compared to routing every query through a massive generalist LLM (Erdogan et al., 2024).
- **Token Optimization per Turn:** Because the SRM's context is completely isolated from the main LLM's conversational synthesis, the final generated response (which can be several hundred tokens long) is never appended to the SRM's conversational history. This limits the growth of the input context window for the SRM in multi-turn interactions.

To formalize the decoupled paradigm, Algorithm 1 in Appendix A details the step-by-step execution flow of the IFFC framework. When a query is initiated, the SRM evaluates the intent using PFC instead of NFC. If tool execution is deemed necessary, the system executes the function and appends the resulting context directly to the query.

3.2 Instruction-Followed Function Calling (IFFC)

Our second major finding is that models, especially SRMs, demonstrate higher accuracy when performing tool selection within a standard "Instruction-Following" context rather than a specialized "Tool Calling" context. Most modern LLMs are trained with native function-calling (NFC) support, requiring specific tags (e.g., `<tool_declare>` and `<tool_call>`) and rigid JSON schemas. However, empirical results from the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025) and our own experiments suggest that these rigid constraints often lead to formatting errors and hallucinations in smaller models (Kavathekar et al., 2025; Johnson et al., 2025); Which eventually shows that LLMs are more capable of function calling when they are prompted, rather than when they are declared in special tool tokens.

We introduce **Instruction-Followed Function Calling (IFFC)**, which bypasses the native API-call wrappers in favor of natural language instructions. As shown in the comparison in our framework's prompt structure:

- **Native Function Calling (NFC):** Uses specialized, non-conversational tokens to declare

Table 1: Impact of Q4KM Quantization on Memory Usage and IFFC Accuracy (BFCL V3 Non-Live)

	Memory Footprint			Accuracy Degradation			
Model	**FP16**	**Q4KM**	**Reduction**	**FP16**	**Q4KM**	**Δ (pts)**	**Rel. Drop**
Gemma-3 1B	2.0 GB	0.8 GB	60.0%	23.9%	8.7%	–15.2	63.6%
Gemma-3 4B	8.6 GB	3.3 GB	61.6%	81.9%	60.4%	–21.5	26.3%
Gemma-3 12B	24.3 GB	8.1 GB	66.7%	91.3%	69.0%	–22.3	24.4%
Phi-4 Mini	7.7 GB	2.5 GB	67.5%	74.9%	48.2%	–26.7	35.6%
Qwen-3 0.6B (NoThink)	1.5 GB	0.5 GB	66.7%	62.3%	16.8%	–45.5	73.0%
Qwen-3 0.6B (Think)	1.5 GB	0.5 GB	66.7%	76.4%	69.2%	–7.2	9.4%
Qwen-3 1.7B (NoThink)	4.0 GB	1.4 GB	65.0%	82.8%	80.5%	–2.3	2.8%
Qwen-3 1.7B (Think)	4.0 GB	1.4 GB	65.0%	82.8%	82.9%	+0.1	–0.1%
Qwen-3 4B (NoThink)	8.0 GB	2.5 GB	68.8%	91.9%	91.0%	–0.9	1.0%
Qwen-3 4B (Think)	8.0 GB	2.5 GB	68.8%	94.1%	93.5%	–0.6	0.6%
Qwen-3 8B (NoThink)	16.4 GB	5.2 GB	68.3%	93.8%	92.8%	–1.0	1.1%
Qwen-3 8B (Think)	16.4 GB	5.2 GB	68.3%	94.2%	94.4%	+0.2	–0.2%

Table 2: Average Function Calling Accuracy (%) Over Four Categories on BFCL V3

Model	**IFFC (Ours)**	**PFC**	**NFC**
BFCL V3 Live (Average)			
Phi-4 Mini	44.3	**62.5**	30.0
Qwen-3 0.6B Think	**57.7**	53.2	53.0
Qwen-3 1.7B Think	72.8	73.6	**74.6**
Qwen-3 4B Think	**86.7**	82.8	81.5
Qwen-3 8B Think	**83.3**	78.3	75.5
BFCL V3 Non-Live (Average)			
Phi-4 Mini	**77.4**	41.9	9.5
Qwen-3 0.6B Think	**76.5**	72.8	71.8
Qwen-3 1.7B Think	**84.0**	65.3	83.0
Qwen-3 4B Think	**94.1**	88.7	88.6
Qwen-3 8B Think	**94.9**	89.7	88.8

and call tools, which can be brittle and sensitive to formatting errors.

- **IFFC Paradigm:** Treats the tool definition as a high-priority system instruction and the function call as a standard assistant response.

By framing the task as a regular user-assistant interaction, we leverage the extensive instruction-tuning that these models undergo. Instead of the SRM struggling with the syntactic overhead of NFC, it follows a system prompt that explicitly defines the tools as part of its "behavioral guidelines." This transition from *manipulating rigid definitions* to *following conversational instructions* allows the SRM to focus its reasoning capacity on argument generation and tool selection logic, leading to the performance gains observed in Tables 2, 3 and 4.

4 Experiments

To validate the efficacy of the IFFC framework, we conducted a comprehensive empirical analysis comparing it against established NFC and PFC paradigms. Our experimental design focuses on three key dimensions: the comparative performance of SLMs, specially SRMs, against state-of-the-art proprietary models, the impact of "thinking" modes in hybrid reasoning architectures, and the robustness of our approach under quantization for edge deployment.

We benchmarked our method using a diverse suite of open-weights models to represent the landscape of efficient SRMs. Specifically, we evaluated the **Gemma-3** series (1B, 4B, and 12B) (Team et al., 2025), **Phi-4 Mini Instruct** (Abouelenin et al., 2025), and the **Qwen-3** series (0.6B, 1.7B, 4B, and 8B) (Yang et al., 2025). Additionally, to test the limits of extreme compression, we included **Granite 4 Micro** and **Granite 4 Tiny-h** (Granite Team, 2024) (detailed specifications provided in the Appendix C).

To establish a rigorous baseline, we compared these SRMs operating under IFFC against the current state-of-the-art proprietary models operating in their native function calling (NFC) modes. These baselines include **GPT 5.2**, **Gemini 2.5 Pro** (Comanici et al., 2025), and **Claude 4.5 Sonnet**. This comparison aims to determine if decoupled SRMs can rival the performance of massive generalist models in tool-use scenarios.

To isolate the effect of reasoning, we utilized the **Qwen-3** (Yang et al., 2025) hybrid models, which support toggleable inference modes. This ablation study allows us to quantify how much the explicit reasoning trace contributes to accurate argument parsing and schema adherence compared to standard generation.

For IFFC to be a viable "plug-and-play" solution for edge AI, it must maintain performance when

Table 3: Model Accuracy (%) Comparison on BFCL V3 Live and Non-Live Benchmarks

	BFCL V3 Live				BFCL V3 Non-Live			
Model	**Simple**	**Multiple**	**Parallel**	**Par. Mult.**	**Simple**	**Multiple**	**Parallel**	**Par. Mult.**
Qwen-3 4B Think IFFC (Ours)	**90.3**	**81.5**	**87.5**	**87.5**	**96.0**	**97.5**	92.5	90.5
Qwen-3 4B Think NFC	87.6	79.9	75.0	83.3	75.3	96.5	92.0	90.5
Claude 4.5 Sonnet NFC	89.5	78.9	**87.5**	83.3	72.6	95.5	**94.5**	**92.0**
GPT 5.2 NFC	71.7	70.4	68.8	58.3	72.9	88.0	89.0	77.5
Gemini 2.5 Pro NFC	77.9	62.2	68.8	62.5	66.4	86.0	69.0	40.0

models are compressed. We conducted a sensitivity analysis by comparing the performance of our selected models at **FP16** versus **Q4KM** quantization levels.

Given that IFFC shares architectural similarities with PFC, in that both rely on natural language prompts rather than specialized tokens, we performed a direct comparison between the two methods using the **Gemma-3** (Team et al., 2025) model family. Full details of the differences between IFFC and PFC is discussed in Appendix A

5 Results

5.1 Superiority of Instruction Following over Native Tool Use

Our primary hypothesis was that SRMs perform better when tool execution is framed as a conversational instruction rather than a rigid schema constraint. The results presented in Tables 3 and 2 strongly corroborate this.

Across the Qwen-3 series (0.6B to 8B) and Phi-4 Mini, IFFC consistently outperforms the NFC baseline. The disparity is particularly pronounced in smaller models; for instance, Phi-4 Mini achieves only $\sim$30% accuracy in NFC mode (Live) but jumps to **44.3%** using IFFC. Similarly, the Qwen-3 4B (Think) model sees a substantial improvement, reaching **86.7%** in Live evaluation and **94.1%** in Non-Live evaluation, significantly surpassing both the PFC (82%) and NFC (81%) baselines.

We further analyzed the limitations of PFC using the Gemma-3 family. As shown in Table 4, while PFC performs adequately on "Simple" queries, it suffers catastrophic degradation in complex scenarios. In the "Parallel Multiple" category, Gemma-3 4B using PFC drops to near 0% accuracy due to context drift and hallucination. In contrast, the IFFC framework maintains robustness, with the Gemma-3 12B model achieving **79.1%** accuracy in the same category. This confirms that decoupling the routing logic allows models to handle high-complexity queries without the context pollution inherent in standard prompting methods.

5.2 Small Reasoning Models vs. Proprietary Giants

The results, illustrated in Table 3, demonstrate that our decoupled SRM approach is highly competitive. In the "Non-Live" evaluation, Qwen-3 4B (IFFC) achieves **96.0%** on Simple tasks and **97.5%** on Multiple tasks, outperforming Claude 4.5 Sonnet (72.6% and 95.5% respectively) and GPT-5.2 (72.9% and 88.0% respectively). Even in the challenging "Parallel Multiple" category, Qwen-3 4B achieves **90.5%**, which is comparable to Claude 4.5 Sonnet (92.0%) and significantly higher than Gemini 2.5 Pro (40.0%). This indicates that a specialized 4B parameter model, when relieved of the syntactic burden of native API definitions, can match or exceed the reasoning fidelity of models orders of magnitude larger.

5.3 The Impact of Reasoning ("Thinking") on Tool Selection

Table 5 presents the comparison between "Think" (Reasoning enabled) and "No-Think" (Standard generation) modes under the IFFC framework.

The reasoning effect is most dramatic in the smallest models; Qwen-3 0.6B sees its accuracy arguably double, jumping from 22.9% to **57.7%** in Live evaluation when reasoning is enabled. For the 4B model, the "Think" mode pushes accuracy from 74.6% to **86.7%**.

5.4 Robustness to Quantization

The results in Table 1 reveal a critical divergence between standard instruction models and reasoning models.

Standard models suffer significant degradation under quantization; for example, Gemma-3 1B drops from 23.9% to **8.7%**, and the standard Qwen-3 0.6B (No-Think) drops from 62.3% to **16.8%**. However, reasoning-oriented models exhibit re-

Table 4: Comparison of Gemma-3 Models using IFFC vs. PFC across BFCL V3 Live and Non-Live Categories.

Model & Method	Simple (%)	Multiple (%)	Parallel (%)	Par. Mult. (%)
BFCL V3 Live				
Gemma-3 1B IFFC (Ours)	13.9	7.1	**31.3**	**8.3**
Gemma-3 1B PFC	**30.0**	**10.5**	0.0	0.0
Gemma-3 4B IFFC (Ours)	**77.1**	**63.7**	**75.0**	**54.2**
Gemma-3 4B PFC	72.9	62.8	37.5	29.2
Gemma-3 12B IFFC (Ours)	**86.4**	**78.5**	**87.5**	**79.2**
Gemma-3 12B PFC	84.9	70.9	**87.5**	62.5
BFCL V3 Non-Live				
Gemma-3 1B IFFC (Ours)	21.8	36.0	**22.0**	**16.0**
Gemma-3 1B PFC	**43.5**	**38.5**	2.0	2.0
Gemma-3 4B IFFC (Ours)	**87.6**	85.0	**82.5**	**72.5**
Gemma-3 4B PFC	64.3	**91.5**	56.5	41.0
Gemma-3 12B IFFC (Ours)	**94.0**	93.5	**90.0**	**89.0**
Gemma-3 12B PFC	77.3	**95.0**	**90.0**	73.0

Table 5: Performance Comparison of Qwen-3 in No-Think vs. Think Modes under the IFFC Framework

	BFCL V3 Live (%)		BFCL V3 Non-Live (%)	
Model	**No-Think**	**Think**	**No-Think**	**Think**
Qwen-3 0.6B	22.9	**57.7**	62.3	**76.5**
Qwen-3 1.7B	65.0	**72.8**	81.1	**84.0**
Qwen-3 4B	74.6	**86.7**	91.9	**94.1**
Qwen-3 8B	74.5	**83.3**	93.8	**94.9**

markable resilience. The Qwen-3 4B (Think) model maintains **93.5%** accuracy at Q4KM, a negligible drop from 94.1% at FP16. Similarly, the 8B (Think) model actually shows a slight variance improvement to 94.4%.

This finding suggests that the "reasoning trace" acts as a form of error correction that compensates for the precision loss in model weights, making SRMs uniquely working for efficient on-device function calling. Full experiment results are presented in Appendix C

6 Conclusion

In this work, we introduced Instruction-Followed Function Calling (IFFC), a framework that fundamentally redefines how Small Reasoning Models (SRMs) execute agentic tasks by prioritizing natural language instruction adherence over rigid native tool definitions. Our extensive empirical evaluation demonstrates that decoupling the routing logic enables compact models, such as the Qwen-3 4B, to outperform massive proprietary baselines like GPT-5.2 and Claude 4.5 Sonnet, particularly when leveraging explicit reasoning traces. Furthermore, we validated the remarkable robustness of reasoning-oriented models under aggressive quantization, confirming their viability for efficient, privacy-preserving edge deployment.

7 Limitations

While the Instruction-Followed Function Calling (IFFC) framework demonstrates significant accuracy improvements and robustness under quantization, the decoupled two-stage architecture introduces inherent latency and scalability trade-offs. By delegating intent evaluation and tool routing to a dedicated Small Reasoning Model (SRM) before passing the enriched context to the primary QA model, the system necessitates sequential inference steps. Although aggressively quantizing the SRM to Q4KM mitigates memory footprint and computational overhead, real-time applications with strict latency budgets may still experience delays.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, and 1 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Richard Charles 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.

IBM Granite Team. 2024. Granite 3.0 language models. *URL: https://github. com/ibm-granite/granite-3.0-language-models*.

Dongge Han, Menglin Xia, Daniel Madrigal Diaz, Samuel Kessler, Ankur Mallick, Xuchao Zhang, Mirian Del Carmen Hipolito Garcia, Jin Xu, Victor Rühle, and Saravan Rajmohan. 2025. Enhancing reasoning capabilities of small language models with blueprints and prompt template search. *arXiv preprint arXiv:2506.08669*.

Shengtao He. 2024. Achieving tool calling functionality in llms using only prompt engineering without fine-tuning. *arXiv preprint arXiv:2407.04997*.

Xinyu Huang, Leming Shen, Zijing Ma, and Yuanqing Zheng. 2026. Towards privacy-preserving and personalized smart homes via tailored small language models. *IEEE Transactions on Mobile Computing*.

Changhyun Jeon, Jinhee Park, Jungwoo Choi, Keonwoo Kim, Jisu Kim, and Minji Hong. 2025. Slm-based agentic ai with pcg: Optimized for korean tool use. *arXiv preprint arXiv:2509.19369*.

Polaris Jhandi, Owais Kazi, Shreyas Subramanian, and Neel Sendas. 2025. Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning. *arXiv preprint arXiv:2512.15943*.

Hong Jia, Shiya Fu, Feng Xia, Vassilis Kostakos, and Ting Dang. 2025. Beyond scale: Small language models are comparable to gpt-4 in mental health understanding. *arXiv preprint arXiv:2507.08031*.

Reid T Johnson, Michelle D Pain, and Jordan D West. 2025. Natural language tools: A natural language approach to tool calling in large language agents. *arXiv preprint arXiv:2510.14453*.

Ishan Kavathekar, Raghav Donakanti, Ponnurangam Kumaraguru, and Karthik Vaidhyanathan. 2025. Small models, big tasks: An exploratory empirical study on small language models for function calling. In *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering*, pages 1117–1126.

Yahya Sowti Khiabani, Farris Atif, Chieh Hsu, Sven Stahlmann, Tobias Michels, Sebastian Kramer, Benedikt Heidrich, M Saquib Sarfraz, Julian Merten, and Faezeh Tafazzoli. 2025. Optimizing small language models for in-vehicle function-calling. *arXiv preprint arXiv:2501.02342*.

Aman Kumar, Ekant Muljibhai Amin, Xian Yeow Lee, Lasitha Vidyaratne, Ahmed K Farahat, Yuta Koreeda, and Chetan Gupta. 2025. Building domain-specific small language models on a shoestring via guided data generation. In *Large Language Models for Scientific and Societal Advances*.

Souvik Nath, Sumit Wadhwa, and Luis Perez. 2025. Domain-adaptive small language models for structured tax code prediction. *arXiv preprint arXiv:2507.10880*.

Dhruvi Paprunia, Vansh Kharidia, and Pankti Doshi. 2025. Advancing slm tool-use capability using reinforcement learning. In *2025 IEEE 4th World Conference on Applied Intelligence and Computing (AIC)*, pages 92–97. IEEE.

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 *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, and 1 others. 2025. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *arXiv preprint arXiv:2506.14728*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024. From exploration to mastery: Enabling llms to master tools via self-driven interactions. *arXiv preprint arXiv:2410.08197*.

Nicholas Roth, Christopher Hidey, Lucas Spangher, William F Arnold, Chang Ye, Nick Masiewicki, Jinoo Baek, Peter Grabowski, and Eugene Ie. 2025. Factored agents: Decoupling in-context learning and memorization for robust tool use. *arXiv preprint arXiv:2503.22931*.

Ivan Samoylenko. Position: The field of small language models needs greater attention and a more systematic approach from the cs research community. In *ICML 2025 Workshop on Machine Learning for Wireless Communication and Networks (ML4Wireless)*.

Raghav Sharma and Manan Mehta. 2025. Small language models for agentic systems: A survey of architectures, capabilities, and deployment trade offs. *arXiv preprint arXiv:2510.03847*.

Neelabh Sinha, Vinija Jain, and Aman Chadha. 2025. Are small language models ready to compete with large language models for practical applications? In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 365–398.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, and Julian McAuley. 2024. Small models are valuable plug-ins for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 283–294.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Guancheng Zeng, Xueyi Chen, Jiawang Hu, Shaohua Qi, Yaxuan Mao, Zhantao Wang, Yifan Nie, Shuang Li, Qiuyang Feng, Pengxu Qiu, and 1 others. 2025. Routine: A structural planning framework for llm agent system in enterprise. *arXiv preprint arXiv:2507.14447*.

A IFFC and PFC: A review on differences

A.1 The Necessity of Prompting

Not all Large Language Models (LLMs) are trained with native function-calling capabilities. Several high-performance open-weights models, such as the **Gemma-3** (Team et al., 2025) family, do not utilize specialized control tokens (e.g., `<tool_call>`) or separate API-calling heads. To enable function calling in these architectures, we utilize Prompt-based Function Calling (PFC).

In this paradigm, we inject a robust system prompt that defines the model's persona as an expert in function execution. The available tools are serialized into the system context, and the model is instructed to invoke them when necessary. Crucially, unlike Native Function Calling (NFC) which outputs to a specialized `tool_use` role, PFC models generate the function invocation directly within the standard `assistant` response role.

A.2 Relation to IFFC

Our Instruction-Followed Function Calling (IFFC) framework shares the fundamental nature of PFC: it treats tool usage as a text-generation task governed by strict instruction adherence. However, IFFC diverges from standard PFC baselines (such as those found in the BFCL repository) in two key ways:

1. **Decoupled Architecture:** Standard PFC usually feeds the tool definitions to the main conversational model. IFFC offloads this entirely to a dedicated Small Reasoning Model (SRM), isolating the context.

2. **Customized Formatting:** While standard benchmarks often demand a generic JSON format, IFFC enforces a specialized output format optimized for the SRM's reasoning capabilities.

A.3 Targeted Prompt Engineering and Heuristics

Through iterative testing on the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025), we identified specific weaknesses where even capable models struggled with rigid evaluation criteria. To address this, we integrated targeted hints into the IFFC system prompt to guide the SRM's logic.

A primary example involves parameter formatting. We observed that models often hallucinated argument formats that were semantically correct but syntactically mismatched with the API definition (e.g., providing a city name alone when the API required a specific string pattern). We explicitly updated the system instruction to force adherence to examples provided in the function docstrings.

For instance, for a function `get_weather(location: str)`, if the documentation provides an example value like `"City, Country"`, our prompt explicitly instructs the model to: *"Use the exact same format of function parameter values as the examples in the function definition."* This ensures that the model generates `"Paris, France"` rather than just `"Paris"`, significantly reducing schema validation errors during evaluation.

B IFFC System Prompt

To ensure reproducibility, we provide the exact system prompt utilized in the Instruction-Followed Function Calling (IFFC) framework. The `{available_functions}` placeholder is dynamically populated with the specific tool definitions relevant to the query.

Algorithm 1 Instruction-Followed Function Calling (IFFC) Workflow

Require: User Query Q, Available Functions $\mathcal{F}$, SRM System Prompt P_{SRM} (with IFFC guidelines), Main LLM System Prompt P_{Main}, SRM Conversational State $\mathcal{H}_{\text{SRM}}$, Main LLM Conversational State $\mathcal{H}_{\text{Main}}$

Ensure: Final Conversational Response R

// Stage 1: Intent evaluation and function routing via SRM
Construct input payload for SRM: $X_{\text{SRM}} \leftarrow \text{FormatPrompt}(P_{\text{SRM}}, \mathcal{F}, Q, \mathcal{H}_{\text{SRM}})$
Generate SRM response: $O_{\text{SRM}} \leftarrow \text{SRM}(X_{\text{SRM}})$ {Executed via Prompt-based Function Calling (PFC)}

// Analyze output for required external knowledge
if O_{SRM} triggers a function call $f \in \mathcal{F}$ with parameters θ **then**
    Execute external function: $C \leftarrow \text{Execute}(f, \theta)$ {Retrieve external context}
    Formulate context-enriched query: $Q_{\text{enriched}} \leftarrow \text{Combine}(Q, C)$
else
    $Q_{\text{enriched}} \leftarrow Q$
end if

// Stage 2: Independent Response Synthesis via Main QA LLM
Construct input payload for Main LLM: $X_{\text{Main}} \leftarrow \text{FormatPrompt}(P_{\text{Main}}, Q_{\text{enriched}}, \mathcal{H}_{\text{Main}})$
Generate final synthesized response: $R \leftarrow \text{LLM}_{\text{Main}}(X_{\text{Main}})$

// Stage 3: State separation and token reduction update
Update Main LLM memory: $\mathcal{H}_{\text{Main}} \leftarrow \mathcal{H}_{\text{Main}} \cup \{Q, R\}$
Update SRM memory: $\mathcal{H}_{\text{SRM}} \leftarrow \mathcal{H}_{\text{SRM}} \cup \{Q, O_{\text{SRM}}\}$ {Excludes R to preserve the SRM's context space}

return R

IFFC System Instruction

```
 You are an expert in function calling. You will be given a set of available functions and a
user query. You should determine if the user query needs function calling, call the appropriate
function with suitable arguments. Based on the question , you will need to make one or more
function calls to achieve the purpose.

Here are the available functions:
{available_functions}

For every user query you only answer in json format. You will return a list of dictionaries. If
the user question doesn't require any function calling you will return an empty list ( e.g. []
). If it does require function calling, for every function you add a dictionary which has two
keys, "function_name" and "parameters". "parameters" is also a dict with function arguments
and their values. Use the exact same format of function parameter values as the examples in the
function definition.
```

B.1 Decoupling in IFFC

One of the core differences between traditional function calling and IFFC is the context separation of the QA LLM and the function caller, which leads the function caller to be less context polluted. In Algorithm 1 we provide a detailed schema of how context separation operates in IFFC.

C Full Results

The following tables present the comprehensive results of our experiments on the Berkeley Function Calling Leaderboard (BFCL). We compare our Instruction-Followed Function Calling (IFFC) framework (in both FP16 and Q4KM quantization) against Prompt Function Calling (PFC) and Native Function Calling (NFC).

Notes:

- **Gemma-3** models do not support NFC; these entries are marked as (-).
- **Granite-4** models were primarily evaluated in Q4KM due to resource constraints.
- In the *Live Parallel* and *Live Parallel Multiple* categories, IFFC FP16 results are omitted for most models due to time constraints during the evaluation window.
- The IFFC columns are highlighted in gray for clarity.

Table 6: Non-Live Evaluation: Simple and Multiple Categories

	Non-Live Simple				**Non-Live Multiple**			
Model	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**
Gemma-3 1B	21.8%	3.6%	43.5%	-	36.0%	21.5%	38.5%	-
Gemma-3 4B	87.6%	87.5%	64.3%	-	85.0%	84.0%	91.5%	-
Gemma-3 12B	93.2%	94.0%	77.3%	-	93.0%	93.5%	95.0%	-
Phi-4 Mini	62.7%	72.7%	67.9%	38.0%	76.5%	63.5%	69.0%	0.0%
Granite-4 Micro	-	81.2%	-	-	-	82.0%	-	-
Granite-4 Tiny H	-	46.5%	-	-	-	31.5%	-	-
Qwen-3 0.6B (NoThink)	53.7%	4.2%	-	-	68.0%	19.0%	-	-
Qwen-3 0.6B (Think)	82.0%	74.5%	64.0%	62.3%	89.5%	76.5%	89.0%	88.0%
Qwen-3 1.7B (NoThink)	87.2%	84.5%	-	-	78.5%	81.5%	-	-
Qwen-3 1.7B (Think)	80.1%	80.5%	-	71.1%	81.0%	84.5%	92.5%	93.0%
Qwen-3 4B (NoThink)	94.0%	93.5%	-	-	95.5%	94.0%	-	-
Qwen-3 4B (Think)	96.0%	95.5%	76.1%	75.3%	97.5%	96.5%	97.0%	96.5%
Qwen-3 8B (NoThink)	96.0%	95.7%	-	-	96.0%	96.0%	-	-
Qwen-3 8B (Think)	96.5%	95.7%	78.4%	76.8%	97.5%	96.5%	96.0%	95.5%

Table 7: Non-Live Evaluation: Parallel and Parallel Multiple Categories

	Non-Live Parallel				Non-Live Parallel Multiple			
Model	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**
Gemma-3 1B	22.0%	-	2.0%	-	16.0%	9.5%	2.0%	-
Gemma-3 4B	82.5%	-	56.5%	-	72.5%	70.0%	41.0%	-
Gemma-3 12B	90.0%	-	90.0%	-	89.0%	88.5%	73.0%	-
Phi-4 Mini	78.5%	-	16.0%	0.0%	82.0%	56.5%	14.5%	0.0%
Granite-4 Micro	-	80.0%	-	-	-	78.5%	-	-
Granite-4 Tiny H	-	70.5%	-	-	-	71.5%	-	-
Qwen-3 0.6B (NoThink)	65.0%	18.5%	-	-	62.5%	25.5%	-	-
Qwen-3 0.6B (Think)	69.0%	69.5%	75.0%	69.0%	65.0%	56.5%	63.0%	68.0%
Qwen-3 1.7B (NoThink)	85.0%	77.5%	-	-	80.5%	78.5%	-	-
Qwen-3 1.7B (Think)	89.0%	84.5%	88.0%	87.5%	81.0%	82.0%	81.5%	81.0%
Qwen-3 4B (NoThink)	89.5%	89.5%	-	-	88.5%	87.0%	-	-
Qwen-3 4B (Think)	92.5%	92.0%	92.0%	92.0%	90.5%	90.0%	89.5%	90.5%
Qwen-3 8B (NoThink)	93.5%	91.0%	-	-	89.5%	88.5%	-	-
Qwen-3 8B (Think)	93.5%	95.0%	95.0%	94.5%	89.5%	90.5%	89.5%	88.5%

Table 8: Live Evaluation: Simple and Multiple Categories

	Live Simple				Live Multiple			
Model	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**
Gemma-3 1B	13.9%	8.8%	30.0%	-	3.1%	7.1%	10.5%	-
Gemma-3 4B	72.4%	77.1%	72.9%	-	63.5%	63.7%	62.8%	-
Gemma-3 12B	85.2%	86.4%	84.9%	-	74.5%	78.5%	70.9%	-
Phi-4 Mini	42.0%	39.0%	55.0%	40.3%	-	47.6%	59.5%	50.3%
Granite-4 Micro	-	68.9%	-	-	-	60.4%	-	-
Granite-4 Tiny H	-	31.7%	-	-	-	35.1%	-	-
Qwen-3 0.6B (NoThink)	50.0%	3.1%	-	-	29.1%	2.0%	-	-
Qwen-3 0.6B (Think)	66.2%	58.5%	66.1%	65.9%	52.2%	47.5%	52.2%	54.4%
Qwen-3 1.7B (NoThink)	66.6%	58.1%	-	-	53.1%	60.2%	-	-
Qwen-3 1.7B (Think)	75.9%	71.7%	75.6%	75.6%	68.6%	73.5%	68.6%	72.6%
Qwen-3 4B (NoThink)	78.6%	75.1%	-	-	73.8%	73.4%	-	-
Qwen-3 4B (Think)	90.3%	86.4%	87.9%	87.6%	81.5%	79.9%	80.7%	79.9%
Qwen-3 8B (NoThink)	73.6%	73.6%	-	-	78.6%	76.0%	-	-
Qwen-3 8B (Think)	87.9%	87.2%	87.2%	84.9%	80.4%	80.7%	79.4%	79.4%

Table 9: Live Evaluation: Parallel and Parallel Multiple Categories

	Live Parallel				Live Parallel Multiple			
Model	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**	**IFFC FP16**	**IFFC Q4KM**	**PFC**	**NFC**
Gemma-3 1B	31.3%	0.0%	0.0%	-	8.3%	0.0%	0.0%	-
Gemma-3 4B	68.8%	75.0%	37.5%	-	54.2%	49.9%	29.2%	-
Gemma-3 12B	87.5%	87.5%	87.5%	-	79.2%	79.2%	62.5%	-
Phi-4 Mini	-	50.0%	68.8%	0.0%	-	37.5%	66.7%	29.2%
Granite-4 Micro	-	75.0%	-	-	-	-	-	-
Granite-4 Tiny H	-	18.8%	-	-	-	-	-	-
Qwen-3 0.6B (NoThink)	-	0.0%	-	-	-	12.5%	-	-
Qwen-3 0.6B (Think)	-	62.5%	40.3%	37.5%	-	50.0%	54.2%	54.2%
Qwen-3 1.7B (NoThink)	-	62.5%	-	-	-	70.8%	-	-
Qwen-3 1.7B (Think)	-	75.0%	75.0%	75.0%	-	66.6%	75.0%	75.0%
Qwen-3 4B (NoThink)	-	75.0%	-	-	-	70.8%	-	-
Qwen-3 4B (Think)	-	87.5%	77.5%	75.0%	-	87.5%	85.0%	83.3%
Qwen-3 8B (NoThink)	-	75.0%	-	-	-	70.8%	-	-
Qwen-3 8B (Think)	-	81.3%	67.3%	62.5%	-	83.3%	79.2%	75.0%

Table 10: Proprietary Models Baseline (Native Function Calling)

	General Purpose Models							Claude Series	
Category	**GPT 4.1 Mini**	**GPT 4.1**	**Gemini 2.5 Pro**	**Grok 4 07-09**	**Kimi K2 Inst.**	**O4 mini**	**GPT 5.2**	**Claude Opus 4.5**	**Claude Sonnet 4.5**
Simple	73.8%	74.2%	66.4%	73.5%	78.2%	70.6%	72.9%	76.8%	72.6%
Multiple	93.5%	90.5%	86.0%	92.5%	93.0%	84.5%	88.0%	95.5%	95.5%
Parallel	93.0%	91.0%	69.0%	88.5%	85.5%	0.0%	89.0%	93.5%	94.5%
Parallel Multiple	87.0%	86.0%	40.0%	87.0%	84.0%	0.0%	77.5%	88.5%	92.0%
Live Simple	80.6%	80.2%	77.9%	82.2%	88.0%	68.6%	71.7%	86.4%	89.5%
Live Multiple	77.8%	78.4%	62.2%	73.9%	79.4%	68.1%	70.4%	48.2%	78.9%
Live Parallel	75.0%	68.8%	68.8%	75.0%	87.5%	0.0%	68.8%	87.5%	87.5%
Live Par. Multi.	66.7%	66.7%	62.5%	79.2%	62.5%	0.0%	58.3%	75.0%	83.3%