Execution-First Synthetic Tool-Use Traces for Small Language Model Agents

Hafsa Ouajdi[1,2*] **Francesco Giannuzzo**[1,2*] **Alaa Boukhary**[2]
Paolo Papotti[1] Gerard Conangla[2] Adam Elwood[2]
[1]EURECOM, Sophia Antipolis, France
[2]Aily Labs, Munich, Germany
[*]Equal contribution
hafsa.ouajdi@ailylabs.com, Francesco.Giannuzzo@eurecom.fr, alaa.boukhary@ailylabs.com,
papotti@eurecom.fr, gerard.conangla@ailylabs.com, adam.elwood@ailylabs.com

Abstract

Small language models (SLMs) are attractive backbones for agentic systems, but their limited parametric capacity increases their reliance on reliable tool use and, consequently, on training data containing tool interactions that actually execute. Query-first synthesis can produce plausible requests that do not correspond to valid tool sequences, compatible parameters, or available data. We propose SYNTHETICAGENTTRACEQA, an execution-first framework that constructs workflow templates, assigns tools under data-flow constraints, executes and validates traces, and only then synthesizes user tasks, teacher reasoning, and reference answers. Across four tool ecosystems, fine-tuning Qwen3.5-4B and 9B on the resulting data improves tool execution, reference-trace agreement, and answer production (numeric match remains $<10\%$). In our experiments, the best results came from training only on trajectories that executed successfully and supervising tool calls and final answers while masking teacher `<think>` tokens: masked supervision outperformed full supervision on answer production, and full supervision dropped 9B answer production below the base 9B thinking baseline.

1 Introduction

Tool-augmented agents invoke APIs, search, and analytics rather than relying on parametric knowledge alone Yao et al. [2023], Schick et al. [2023], Qin et al. [2023]. This shifts much of the work in an agent loop toward repetitive, narrowly scoped tool invocation, making small language models (SLMs) plausible backbones for agentic deployment. At the same time, SLMs have less capacity to compensate for weak supervision: in multi-step workflows, they may select the wrong tool, emit invalid arguments, ignore observations, or produce answers without execution support Yao et al. [2024], Zhang et al. [2024]. Training them, therefore, requires demonstrations whose tool interactions actually execute, rather than merely demonstrations that look plausible.

Query-first synthesis, via manual scenarios or forward generation from seed instructions Wang et al. [2023], Xu et al. [2025], can yield requests that require unavailable tools, incompatible parameters, or missing data. SYNTHETICAGENTTRACEQA reverses this order: it constructs and executes traces first, retains only successful ones, and then synthesizes user tasks. An operational taxonomy unifies three enterprise ecosystems (finance, research portfolio, supply chain) and ToolBench `genius_song_lyrics` Guo et al. [2024] under one generation process.

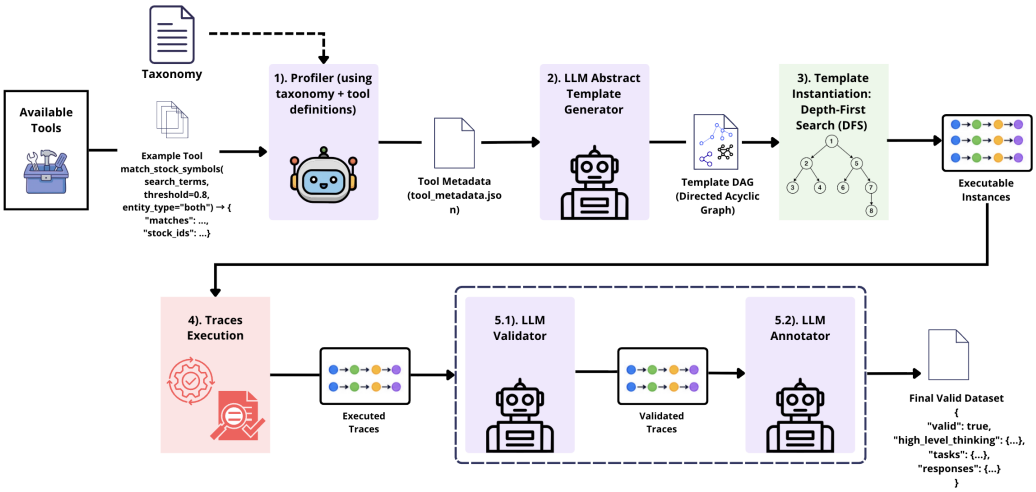


Figure 1: Overview of SYNTHETICAGENTTRACEQA: (1) Profiler, (2) Template Generator, (3) DFS instantiation, (4) Trace Execution, (5) Validation & Task Synthesis.

Research questions. (**RQ1**) Does execution-grounded fine-tuning improve tool behavior, reference-trace agreement, and answer production? (**RQ2**) How do these effects change between 4B and 9B models? (**RQ3**) Does supervising post-hoc rationale tokens help or hurt relative to masking them?

Contributions. (i) An execution-first pipeline with profiler metadata, abstract DAG templates, dependency-aware tool assignment, and runtime validation before task synthesis. (ii) A shared operational taxonomy that supports workflow generation across four heterogeneous tool ecosystems, with ecosystem-specific interface wrappers. (iii) A fine-tuning study on Qwen3.5-4B/9B showing tool-reliability gains and a preliminary Masked-vs-Full comparison (one seed; post-hoc rationales).

Related work. Most tool-use datasets are query-first and can yield invalid call sequences Wang et al. [2023], Xu et al. [2025]. Execution-verified pipelines such as APIGen/xLAM Liu et al. [2024], Zhang et al. [2025] filter synthesized function calls against real APIs, while ToolLLM Qin et al. [2023] searches solution paths after a query is given. Trajectory-centric methods grow or sample workflows with proxy signals such as textual gradients, similarity graphs, or dependency metadata Zhou et al. [2026], Yang et al. [2025], Hao et al. [2026]. Closest to our setting, ToolGrad first grows tool-use chains and then synthesizes queries Zhou et al. [2026]. We instead validate multi-step executions in the target environment *before* any task language is written, and couple this with role-based parameter profiling so compatibility is not reduced to string-matched argument names.

2 The SyntheticAgentTraceQA Pipeline

Let $T = \{T_1, \ldots, T_n\}$ be the available tools, each mapping arguments x_i to an observation y_i. An execution trace is $\tau = [(T_k, x_k, y_k)]_{k=1}^{m}$. We generate tuples (z, τ, r) where τ is executable, the answer r is supported by its observations, and the user task z is a realistic request for that workflow. Figure 1 summarizes the five stages; execution-based filtering occurs *before* any natural-language task is written.

Operational taxonomy. Tools are assigned to one of six classes $\mathcal{O} = \{\text{MATCHER}, \text{RETRIEVER}, \text{ANALYZER}, \text{AGGREGATOR}, \text{METADATA}, \text{ACCESS}\}$ so templates plan over functional roles rather than concrete APIs. Each input or output field is labeled with a role in $\{P_{\text{ctrl}}, P_{\text{flow}}, P_{\text{time}}, P_{\text{set}}\}$ (control, data-flow, temporal, configuration). Two fields are compatible only if their roles, types, and profiler source mappings agree; parameter *names* alone are never used as a match key.

Profiler. Given raw tool schemas, an LLM emits, for every tool T_i: (i) an operational class $o_i \in \mathcal{O}$; (ii) role and type annotations for each parameter and output field; (iii) an *active domain* of feasible

values obtained by sampling the underlying data store or reading enumerated options from the schema; and (iv) explicit upstream→downstream field mappings used later for compatibility checks. The resulting catalog $\mathcal{M}^*(T_i)$ is the only tool metadata consumed by later stages.

Templates and Instantiation. Conditioned on $\mathcal{O}$, $\mathcal{M}^*$, and a target complexity

$$|V| \in \{1,2\} \text{ (easy)}, \quad |V| \in \{3,4\} \text{ (medium)}, \quad |V| \in \{5,6\} \text{ (hard)},$$

an LLM emits an abstract DAG $G = (V, E)$ whose nodes are operational blocks $b_\ell = (o_\ell, \mathrm{pool}_\ell, \mathrm{dep}_\ell, \mathrm{desc}_\ell)$: a class, a pool of compatible tools from $\mathcal{M}^*$, predecessor dependencies, and a short domain description for later task synthesis. Templates must satisfy data-flow consistency (every required P_{flow} input is provided by a predecessor), non-empty tool pools, and the requested node budget.

Instantiation replaces each block with a concrete tool via DFS over pools in topological order. The search maintains a visible state V_{visible} of available output fields. A candidate $T_i \in \mathrm{pool}_\ell$ is admissible iff $P_{\text{flow}}(T_i) \subseteq V_{\text{visible}}$ under the profiler's role/type/source mappings; on success, $\mathrm{OutputFields}(T_i)$ is unioned into V_{visible} and the search continues; otherwise, it backtracks. Argument *values* are not chosen here; only tool identities are. Example: MATCHER→RETRIEVER→ANALYZER may become `match_company`→`retrieve_financial_statement`→`calculate_growth`.

Execution. Each instantiated candidate is run in topological order against the tool environment. A context $C = (C_{\text{in}}, C_{\text{out}}, W)$ stores bound inputs, prior outputs, and temporal constraints. At step k, missing P_{ctrl}/P_{time}/P_{set} values are sampled from the active domain (with optional re-sampling and relaxed matcher thresholds on failure), P_{flow} values are copied from C_{out}, the tool is invoked, and C is updated. A candidate is retained iff *every* call (i) resolves to a registered tool, (ii) passes schema validation, (iii) returns a non-empty usable observation, and (iv) satisfies declared dependencies; otherwise the entire candidate is discarded.

Validation and task synthesis. Only after successful execution does an LLM validator inspect the trace, still without a user question, for DAG consistency, parameter propagation, schema compliance, temporal coherence, and semantic plausibility of the workflow. Failures are discarded. For each surviving trace, the *same* LLM that produced the template writes three user tasks (basic / intermediate / challenging), a teacher rationale linking recorded calls to the answer, and a reference answer r grounded in the observed outputs. Task difficulty controls linguistic underspecification only: basic states intent and entities; intermediate omits some details; challenging is high-level. All three share the same validated τ.

Dataset and domains. Profiler metadata were produced with Claude Sonnet 4.6. Eight LLMs generated templates independently per domain (Claude Sonnet 4.6/4.5, Amazon Nova Pro/Lite, Qwen3-235B-A22B, Gemini 2.5 Pro, Claude Opus 4.1, GPT-5.5), in batches of up to 30 (10 per difficulty) at temperature 0.7; the generating model then validated and synthesized tasks. Overall, 676 templates (287 easy, 282 medium, 107 hard) expanded into 3,203 candidate traces, of which 2,587 (80.8%) passed execution and validation, yielding 7,761 tasks. Fine-tuning uses a 1,000-example split, stratified by trace complexity, with all questions from one trace in the same split. The four ecosystems are finance, research portfolio, supply chain, and ToolBench `genius_song_lyrics`; the latter needed only interface wrappers.

3 Experiments and Results

Setup. We fine-tune Qwen3.5-4B/9B with QLoRA (4-bit NF4 with double quantization; LoRA r=16, α=32, dropout 0.1) on a single NVIDIA L40S (48 GB). Training uses `paged_adamw_8bit`, learning rate 2×10^{-4} with linear decay, batch size 1, gradient accumulation 4, 3 epochs, and seed 42 (one run per configuration). We compare *Masked* supervision, which excludes `<think>` tokens from the loss while supervising tool calls and answers, with *Full* supervision over all assistant tokens. Evaluation uses a multi-turn sandbox (n=200, up to eight greedy turns), executing only post-`</think>` tool calls and returning structured errors for invalid calls. Metrics: answer production (emission rate), token F1 (whitespace token overlap), numeric match (ordered numbers, absolute tolerance 0.01), attempt–success gap Δ=$(\mathrm{Attempt} - \mathrm{Success})/\mathrm{Attempt}$, hallucinated-tool rate,

Table 1: Held-out results (mean ± SE, n=200). Ans. = answer production; F1 = token F1; Δ = attempt–success gap; Hall. = hallucinated-tool rate; Set-F1 = tool-set F1; Seq. = LCS similarity. Masked is strongest on production and reliability; full supervision degrades at 9B.

Model	Ans. (%) ↑	F1 ↑	Δ (%) ↓	Hall. (%) ↓	Set-F1 ↑	Seq. ↑
Base-4B	21.0±2.9	0.04	18.9	18.1	0.43	0.20
4B-Think	41.0±3.5	0.07	6.1	3.2	0.53	0.27
FT-4B-Mask	**59.5±3.5**	**0.15**	**0.23**	**0.23**	**0.57**	**0.34**
FT-4B-Full	35.0±3.4	0.10	0.64	0.50	0.54	**0.34**
Base-9B	25.0±3.1	0.04	19.6	17.3	0.46	0.22
9B-Think	53.5±4.0	0.09	**0.07**	0.08	0.53	0.29
FT-9B-Mask	**86.0±2.5**	**0.14**	**0.07**	**0.07**	**0.57**	**0.38**
FT-9B-Full	17.5±2.7	0.05	0.05	0.06	0.52	0.23

tool-set F1, and LCS sequence similarity. Trace agreement measures similarity to a single reference trajectory rather than unique correctness.

RQ1-RQ2: execution-grounded supervision and scale. Table 1 shows Masked fine-tuning raises answer production (41.0%→59.5% at 4B; 53.5%→86.0% at 9B), cuts 4B Δ from 6.1% to 0.23%, and lifts tool-set F1 to 0.57. The reliability gains are the more convincing result: production measures emission, not correctness, numeric match stays low (best 9.83%) and token F1 ≤ 0.15. Thinking at inference already helps; scale mainly raises production and sequence similarity (0.34 vs. 0.38). For 9B, execution reliability is largely present in the thinking baseline; supervision mainly reduces call count (5.52→4.04).

RQ3: process vs. outcome supervision. Masked supervision outperforms Full on answer production, while FT-9B-Full answer production falls to 17.5%, below the thinking baseline, without consistent gains in trace agreement. Teacher reasoning describes recorded tool calls retrospectively and may be a weak training target Huang et al. [2026]. Supervising executable actions and final answers therefore worked better than full reasoning supervision for answer production and tool reliability in our runs. The gap was largest at 9B. This pattern is consistent with interference between rationale supervision and tool-call learning, although a single seed does not isolate this mechanism from possible `</think>`-formatting or output-format degeneration.

Comparison with Toucan Xu et al. [2025]. Compared with Toucan, our execution-first pipeline produces a higher fraction of accepted traces (80.8% vs. 15%) and lower cost per accepted trace ($0.05 vs. $0.12). An LLM judge also rates our queries higher on realism, clarity, and naturalness, while NovelSum diversity is higher (0.50 vs. 0.39). When Qwen3.5-4B is fine-tuned on each dataset, our data yields larger gains in tool-set F1 (+0.086 vs. +0.061) and sequence similarity (+0.117 vs. +0.086), whereas Toucan yields a larger gain in answer completion (+0.230 vs. +0.195). These comparisons are within-pipeline and use different generation procedures and environments, so they should be interpreted as indicative rather than a matched query-first baseline.

4 Discussion

Implications for SLM agents. Executable traces provide dense, verifiable supervision: Masked fine-tuning reduces the 4B attempt-success gap from 6.1% to 0.23% and substantially improves answer completion. At 9B, where baseline execution is already reliable, gains are concentrated in answer completion and reference-trace alignment. Our strongest evidence is for executable-trace construction and protocol learning, not general answer quality or superiority over query-first synthesis.

Supervision design. In our runs, Masked vs. Full was not a minor hyperparameter choice: full supervision degraded answer production at 9B. In our setting, a better objective was to generate execution-validated trajectories, supervise tool invocations and final answers, and avoid using post-hoc teacher reasoning as a primary SFT target unless it is known to reflect the student's process Huang et al. [2026]. Whether this generalizes beyond Qwen3.5 at 4B/9B, 1,000 training examples, seed 42, and controlled tool environments remains open.

Limitations. Our evaluation uses fixed schemas and controlled environments rather than a shared external agentic benchmark; fine-tuning uses only 1,000 examples and one seed per configuration; answer production is not correctness (numeric match $<10\%$); the generator also writes tasks, rationales, and references; quality judgments for synthetic queries are automated; and the Toucan comparison is within-pipeline, not a matched query-first ablation. Future work will study a controlled query-first baseline, multi-seed training, external benchmarks, training-set scaling, dynamic APIs, and longer-horizon workflows.

5 Conclusion

SYNTHETICAGENTTRACEQA generates supervision from workflows that actually run, writing user tasks only after runtime validation. Fine-tuning on those traces improves tool reliability and reference-trace agreement and raises answer production (with correctness remaining limited). Masking reasoning tokens while supervising tool calls and answers is the stronger objective for the metrics we report; supervising post-hoc rationales can hurt answer production, especially at 9B, in our single-seed setting.

References

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.664. URL https://aclanthology.org/2024.findings-acl.664/.

Bingguang Hao, Zengzhuang Xu, Yuntao Wen, Xinyi Xu, Yang Liu, Tong Zhao, Maolin Wang, Long Chen, Dong Wang, Yicheng Chen, Cunyin Peng, Xiangyu Zhao, Chenyi Zhuang, and Ji Zhang. From failure to mastery: Generating hard samples for tool-use agents, 01 2026.

Zixian Huang, Kaichen Yang, Xu Huang, Feiyang Hao, Qiming Ge, Bowen Li, He Du, Kai Chen, and Qipeng Guo. How to fine-tune a reasoning model? A teacher-student cooperation framework to synthesize student-consistent SFT data. *CoRR*, abs/2604.14164, 2026. doi: 10.48550/ARXIV.2604.14164. URL https://doi.org/10.48550/arXiv.2604.14164.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. Apigen: automated pipeline for generating verifiable and diverse function-calling datasets. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

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. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. Toucan: Synthesizing 1.5m tool-agentic data from real-world mcp environments, 2025.

Chen Yang, Ran Le, Yun Xing, Zhenwei An, Zongchao Chen, Wayne Zhao, Yang Song, and Tao Zhang. Toolmind technical report: A large-scale, reasoning-enhanced tool-use dataset, 11 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024. doi: 10.48550/ARXIV.2406.12045. URL https://doi.org/10.48550/arXiv.2406.12045.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xLAM: A family of large action models to empower AI agent systems. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11583–11597, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.578. URL https://aclanthology.org/2025.naacl-long.578/.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11388–11422, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.637. URL https://aclanthology.org/2024.emnlp-main.637/.

Zhongyi Zhou, Kohei Uehara, Haoyu Zhang, Jingtao Zhou, Lin Gu, Ruofei Du, Zheng Xu, and Tatsuya Harada. ToolGrad: Efficient tool-use dataset generation with textual "gradients". In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 19040–19056, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.950. URL https://aclanthology.org/2026.findings-acl.950/.