Unlocking SLM Agents: Primacy of Initial Planning and Structured Sequentially-Prefilled Efficient Reasoning

Anonymous ACL submission

Abstract

In on-device agentic systems, Small Language Models (SLMs) are essential for privacy-preserving personalization. However, despite their foundational ability to handle individual sub-tasks, they frequently fail at complex orchestration, the process of logically sequencing multiple tool calls and synthesizing their observations to achieve a final goal. In this work, we present **SIPPER** (Structured Initial Planning via Prefilled Efficient Reasoning), a framework designed to unlock the potential of SLM-based on-device agentic systems. First, we identify that while reasoning can be applied at multiple stages of an agentic workflow, such as initial planning, turnwise execution, or final response synthesis, it is most effective at the initial planning stage. Also, although reasoning has been known as an effective test-time scaling to boost the model's capability, purely autonomous reasoning nor overly intricate structured reasoning is effective for SLM-based agentic systems. Instead, we focus on structured logical forcing via expert-guided sequential prefilling within a single reasoning chain. Therefore, **SIPPER** notably boosts the agentic orchestration ability of SLM by the prefilled structured logical forcing in initial planning. Our results demonstrate that **SIPPER** enables SLMs to outperform frontier models such as GPT-4o and Gemini2.5 in complex, real-world scenarios without any additional training costs.

1 Introduction

The evolution of large language models (LLMs) (Achiam et al., 2023; Gemini Team, Google, 2025; Dubey et al., 2024; An Yang, 2023; Jiang et al., 2023) has increasingly shifted toward autonomous agentic systems that utilize external tools adaptively and sequentially, leveraging successive observations, to satisfy complex user requests (OpenClaw, 2026; Claude Code, 2026; Manus AI, 2026). To provide nuanced and personalized services, these agents must access sensitive personal context—including private schedules, financial records, and health data, which necessitates a transition from cloud-based infrastructures to on-device execution to ensure uncompromising data privacy and personalization. Considering the computational constraints of edge devices, small language models (SLMs) have emerged as the central orchestrators for managing multi-step tool-use trajectories (Belcak et al., 2025). While SLMs are capable of formulating a tool call for an individual atomic task of agentic workflow, applying them directly to the complex orchestration of multiple tools for agentic tasks often results in unsatisfactory performance, creating a significant bottleneck for reliable on-device agentic systems.

Reasoning-based test-time scaling (Snell et al., 2025; Wei et al., 2023) can be leveraged as a promising strategy to overcome the limitation of SLMs in agentic tasks; however, unconstrained, open-ended reasoning often incurs prohibitive computational costs and logical instability in SLMs. Also, although various expert-guided reasoning, i.e. structured reasoning, methods have been studied (Yao et al., 2025; Bang et al., 2025; Madaan et al., 2023; Chen et al., 2025, 2024a), they do not consider the constraints of SLMs in complex agentic workflows. Namely, those methods excessively increase the context lengths due to complex reasoning processes that degrade the SLM's performance, or necessitate repetitive re-prefilling escalating computational overhead.

To effectively leverage the expert-guided reasoning considering the resource-constrained edge device environment, we first study where reasoning should be placed within agentic workflows. Specifically, we analyze the effectiveness of applying reasoning at different stages, such as initial planning, turnwise execution, and final response synthesis, as shown in Fig. 1(a). Our analysis reveals that its impact varies significantly across stages and is most

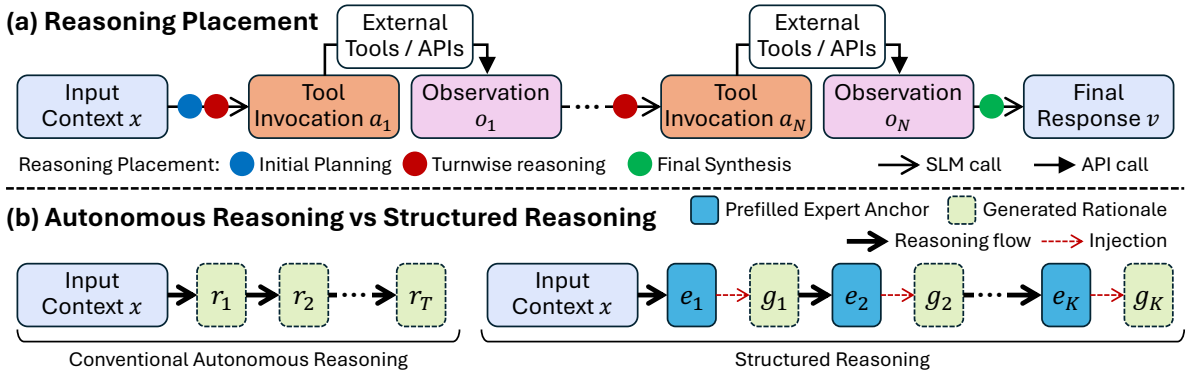

Figure 1: **Reasoning strategies in agentic workflows.** x is input task consisting of the task name and brief one-sentence goal. (a) Candidate placements for applying reasoning within an agentic trajectory. (b) r_t denotes autonomously generated reasoning segments. e_k denotes the injected expert-guided logical anchor (expert anchor), and the g_k represents the generated rationales in response to e_k.

pronounced during the initial planning stage (Section 2.2). Notably, applying reasoning at all stages provides no additional gains and incurs unnecessary overhead for SLMs. Furthermore, we observe that errors introduced at the beginning of the reasoning trajectory are difficult to correct through subsequent turnwise reasoning or final response synthesis, underscoring the importance of establishing a robust logical foundation at the outset.

Then, we propose **SIPPER** (Structured Initial Planning via Prefilled Efficient Reasoning). We anchor the agentic model in expert-guided logical foundation for the initial planning by sequentially-prefilling a pre-defined reasoning structure. While sequentially prefilling multiple expert-guided logical anchors, the model completes the reasoning in response to each logical anchor.

Another significant advantage of **SIPPER** is its synergy with system-level optimizations, particularly in terms of computational efficiency. **SIPPER** avoids frequent context shifts while performing structured reasoning, such as the deletion or re-prefilling of intermediate reasoning steps due to multiple reasoning chains or use of larger models. Namely, by maintaining a single reasoning chain, our approach eliminates computational redundancy and the wasteful re-calculation of hidden states. This allows SLMs to achieve the reasoning accuracy of frontier models with a significantly reduced computational footprint, making it well-suited for resource-constrained edge hardware.

The primary contributions of this paper are summarized as follows:

- We introduce **SIPPER**, which leverages sequential prefilling to outperform frontier LLMs on realistic agentic benchmarks, demonstrating that structured reasoning enables "efficiency-driven scaling" for on-device SLM agents.
- We experimentally demonstrate the primacy of the initial planning stage as the optimal intervention point, proving that structural grounding at the onset is more effective than turnwise modifications.
- **SIPPER** facilitates efficient structured reasoning by maintaining a concise and single reasoning chain, bypassing wasteful re-calculations to enable high-fidelity agentic intelligence on resource-constrained devices.

2 Method

We present **SIPPER** (Structured Initial Planning via Prefilled Efficient Reasoning), a framework that strategically applies structured reasoning to stabilize and scale the performance of the pre-trained SLMs in on-device agentic workflows. Our analysis reveals that among possible reasoning placements, the initial planning stage constitutes the primary bottleneck in multi-step agentic tasks. By utilizing pre-structured sequential logic at this formative stage, we can effectively anchor the agent's logic and establish a robust foundation before the execution loop commences.

2.1 Preliminary: Structured Reasoning

We define the reasoning process as a reasoning path $\mathcal{P}$, which serves as the logical precursor to any ex-

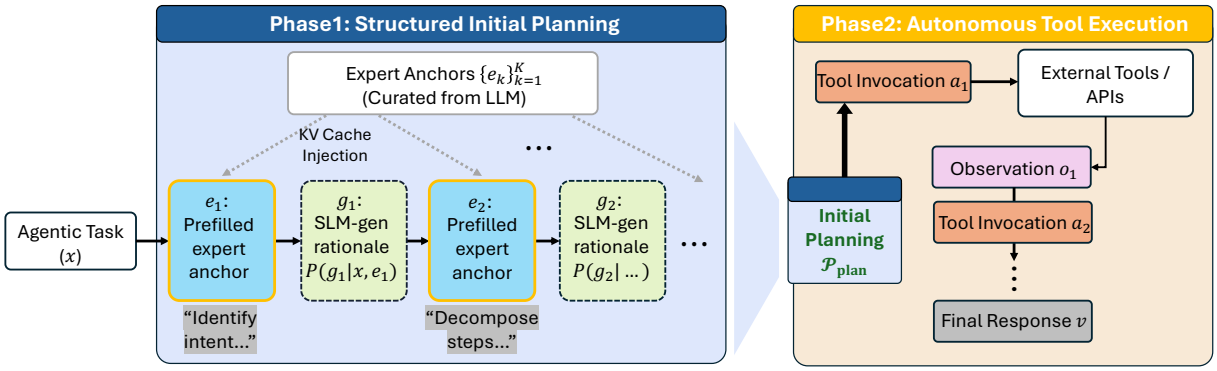

Figure 2: **Overview of SIPPER-based agentic task execution process.** In Phase 1 (left), while the expert anchors e_k are prefilled in sequence, the corresponding rationale g_k are generated to complete the initial planning for x. On top of the planning trajectory, in Phase 2 (right), agent's action a (tool call query generation) and observation o (return from tool call) are repeated until the final response v.

	w/o CoT	Where to CoT			
		Initial Planning	Turnwise	Final Synthesis	All
Acc (%)	62.9	67.3	64.2	64.4	63.8

Table 1: **Importance of reasoning at the initial planning in SLM (Qwen3-4B-Inst).** Average completion (Comp.) accuracy is reported on agentic tasks from the MCPEval benchmark.

ternal action or final response. In conventional autonomous reasoning, the model generates this path $\mathcal{P} = \{r_1, r_2, \dots, r_T\}$ as a continuous sequence of reasoning segments (sentences or phrases) r. The probability of the reasoning path given an input context x is:

$$P(\mathcal{P} \mid x) = \prod_{t=1}^{T} P(r_t \mid x, r_{<t}). \tag{1}$$

In contrast to the recent works that complicate the structured reasoning process, we consider it as simply prefilling pre-defined anchors to trigger generation of corresponding rationale segments, incrementally. Then, as depicted in Fig. 1(b), structured reasoning formulates the reasoning path as the sequence of an interleaved pair of an expert-guided logical anchor, *expert anchor*, (e_k) and a model-generated rationale (g_k) where $k = 1, \dots, K$. Unlike the standard paradigm, the expert anchors e_k are not sampled but are sequentially prefilled to force the reasoning path toward a validated logical structure.

The resulting reasoning path is represented as an ordered sequence: $\mathcal{P}_{\text{struct}} = (e_1, g_1, e_2, g_2, \dots, e_K, g_K)$. The generation of each rationale segment g_k is strictly conditioned on the preceding logical context:

$$P(g_k \mid \text{context}_k) = P(g_k \mid x, \{(e_i, g_i)\}_{i=1}^{k-1}, e_k). \tag{2}$$

Then, the joint probability of the structured reasoning path is represented as:

$$P(\mathcal{P}_{\text{struct}} \mid x) = \prod_{k=1}^{K} P(g_k \mid x, \{e_i, g_i\}_{i=1}^{k-1}, e_k). \tag{3}$$

2.2 Rationale for SIPPER in agentic task

An agentic task is defined by an execution trajectory $\mathcal{T} = (a_1, o_1, a_2, o_2, \dots, a_N, o_N)$, where the model sequentially invokes tools a_j and processes observations o_j. We set the pair of a_n and o_n as nth turn where $n = 1, \dots, N$. In $(N+1)$th turn, the model outputs the final response v.

Primacy of Initial Planning: As depicted in Fig. 1(a), we can consider three candidate stages for structural intervention: (i) at initial planning (before a_1), (ii) in turnwise (before every a_n), and (iii) at final synthesis (between o_N and v). Then, before applying the structured intervention, we apply the naive chain-of-thought (CoT) at the three candidates. As shown in the Table 1, we empirically identified that the initial planning stage is the most significant determinant of task success. From this, we can infer that errors introduced at the onset lead to stochastic drift, which limits the effectiveness of reasoning in the later stage. Additionally, applying CoT at every stage yields diminishing returns for SLMs, where the computational cost of managing extended contexts eventually outweighs and undermines the reasoning gains. From this observation, when refining the reasoning prompts further, we

can attain 78.30% completion accuracy in case (i) while showing a marginal increase in cases (ii) and (iii). Therefore, establishing a robust logical foundation at the very beginning is the most effective way to unlock the full potential of SLM agents.

Limitation of autonomous reasoning in SLM: Also, we checked the average tokens in reasoning from reasoning models (here, Qwen3-4B-Think) is 1.6k even in estimating a single a_n. But its performance gain is only 2.1% in average completion score. Since most of agentic workflow requires multiple a_n's, autonomous reasoning is not practical in the resource-constrained edge devices.

Therefore, we develop **SIPPER** where structured initial planning is introduced by prefilling the predefined logical planning anchors.

2.3 SIPPER

Motivated by the two observations in Section 2.2, Fig. 2 shows the overall workflow of **SIPPER**.

SIPPER Formalization: When applied to the initial planning stage, the agentic process is governed by the coupling of the planning path ($\mathcal{P}_{\text{plan}}$) and the execution trajectory ($\mathcal{T}$). Then, the joint probability is defined as:

$$P(\mathcal{T}, \mathcal{P}_{\text{plan}} \mid x) = \underbrace{\left[\prod_{k=1}^{K} P(g_k \mid x, \{e_i, g_i\}_{i=1}^{k-1}, e_k)\right]}_{\textbf{SIPPER}\text{-Augmented Planning}} \times \underbrace{\prod_{j=1}^{N} P(a_j \mid x, \mathcal{P}_{\text{plan}}, a_{<j}, o_{<j})}_{\text{Autonomous Tool Execution}}. \tag{4}$$

Notice that $\mathcal{T}$ and $\mathcal{P}$ are in a single trajectory without addition of multiple branched trajectories for complex structured reasoning. By simply anchoring the starting point maintaining the single trajectory, **SIPPER** enables SLM agents to resolve complex, multi-step queries reliably, reaching performance level higher than frontier models.

Planning Anchor Design: To avoid the design load of domain-specific anchor logic, **SIPPER** utilizes a domain-agnostic reasoning template. We analyze the agentic task execution results based on SLM on multiple validation domains, and then distilled a universal set of expert anchors. To this end, we utilize the frontier language models. This single, robust template (Fig. 3 in Appendix) allows **SIPPER** to be applied across diverse SLM architectures without manual anchor manipulation.

Algorithm 1: SIPPER: Structured Initial Planning via Prefilled Efficient Reasoning

Input : Input query x, Expert-guided anchors $\mathcal{E} = \{e_1, \ldots, e_K\}$, SLM θ
Output : Final response v, Execution trajectory $\mathcal{T}$

```
// Phase 1: Structured Initial Planning
Initialize planning path P_plan ← ∅, KV Cache C with context x
for k ← 1 to K do
    Prefill anchor e_k into KV Cache C via injection
    Generate rationale segment g_k ~ P(g_k | x, {(e_i, g_i)}_{i=1}^{k-1}, e_k) using θ
    Update planning path P_plan ← P_plan ∪ {e_k, g_k}
    Update C with tokens of g_k
// Phase 2: Autonomous Tool Execution
Initialize trajectory T ← ∅
for j ← 1 to N do
    Generate tool action a_j ~ P(a_j | x, P_plan, a_<j, o_<j)
    if a_j is Final Answer then
        v ← a_j
        break
    Execute a_j and receive observation o_j
    Update trajectory T ← T ∪ {a_j, o_j}
return v, T
```

Efficiency via KV Cache Injection & Reuse: As aforementioned, maintains a monolithic reasoning trajectory from initial planning to the final response. Beyond this structural simplicity, we further optimize execution by leveraging KV cache injection. Instead of generating reasoning rationales autoregressively, each expert anchor e_k is parallelized into a single prefilling forward pass, significantly reducing inference latency. Once computed, these KV caches are preserved and reused in other agentic workflows. Consequently, by bypassing step-by-step token generation and preventing hallucinatory drift through pre-computed states, **SIPPER** achieves reliable and efficient reasoning even in resource-constrained on-device environments. The complete procedure is detailed in Algorithm 1.

3 Experimental Results

3.1 Dataset

We evaluate our system using MCPEval (Liu et al., 2025), a comprehensive, Model Context Protocol (MCP)-based framework designed for the automated generation and deep evaluation of LLM agents. The benchmark encompasses diverse real-world domains: Healthcare, Airbnb, Sports, Book, CRM, National Parks, and Finance. By standardizing agent-environment communication through MCP, this dataset allows for the assessment of complex, multiturn agentic workflows that require sophisticated reasoning and precise tool execution.

To ensure high data quality and reliability, we followed their automated generation and verification pipeline by using two frontier LLMs. Specifically, we utilize GPT-4o as the Task LLM to synthesize the name, goal, description of the agentic task based on MCP tool call list in each domain. To validate each task, we employ Gemini2.5 Pro as a frontier agent to execute the generated tasks and collect execution trajectories. Tasks resulting in execution failure undergo an iterative refinement process, where the agent requests updated descriptions until a verifiable ground truth trajectory is established. Notice that, in testing, we only use the name and goal since the description provide too detailed instruction for ground-truth tool calling.

3.2 Evaluation

Our evaluation adopts LLM-as-a-Judge proposed by MCPEval, which provides a human-centric assessment of complex agent behavior through rubric-based scoring. We utilize a frontier LLM as an expert judge to evaluate two primary categories: trajectory and completion. Trajectory metrics evaluate procedural reasoning, including planning, execution flow, adaptability, and context awareness. Completion metrics measure the quality of the final output, focusing on requirement coverage, factual accuracy, completeness, and practical usefulness. By combining these metrics, we can also capture the execution-completion gap where models may excel at procedural tool use but struggle to synthesize comprehensive final responses.

3.3 Implementation Details

From MCPEval benchmark, to obtain the planning anchors, we set the airbnb, CRM, and sports as the validation domains, and the remaining domains are used in testing. More details for implementation are in Appendix A.

3.4 Baselines

To show the effectiveness of our method, we compare the proposed **SIPPER** with three baselines addressing training-free structured reasoning:

-*Self-Refine* (Madaan et al., 2023) alternates between generating multi-faceted, actionable self-feedback and refining previous outputs with few-shot examples across multiple cycles. It is not tailered for agentic task. Hence, we present two versions: applied at the initial planning only (*planning*) as well as in every turn (*turnwise*).

-*Guide&Refine* (Chen et al., 2025): In a step during structured reasoning, a frontier model detects erroneous reasoning cases which is pre-defined. If detected, a few-shot is given to re-run the step. We also apply two versions (*planning* and *turnwise*) in this baseline

-*InferAct* (Fang et al., 2025) employs two different frontier models to preemtively detect and report the misaligment to the original goals before executing every action. If detected, the expert feedback is given based on the report in every turn.

3.5 Target SLMs

As SLM models for edge-device environment, we mainly utilize Qwen3-4B-Inst (An Yang, 2023) in our experimental analysis. Also, we extend the proposed approach into more diverse SLMs like Qwen3-1.7B (An Yang, 2023) and Qwen3-8B (An Yang, 2023) in Appendix.

3.6 Comparative Results

Table 2 presents a comprehensive evaluation of **SIPPER** against the recent training-free reasoning methods and frontier proprietary models like GPT-4o and Gemini2.5.

SLM Can Beat Frontier LLMs: Our empirical results demonstrate the clear superiority of **SIPPER**, achieving a trajectory accuracy of 86.30% and a completion accuracy of 85.39%, effectively outperforming even the most advanced frontier models, including GPT-4o (79.65% in trajectory and 76.71% in completion) and Gemini 2.5 Pro (82.99% in trajectory and 84.61% in completion), without any additional training or the need for frontier model intervention during inference.

Failure of SLM-led Verification: A critical observation from our experiments is the performance

	Method	Venue	Trajectory (%)	Completion (%)
LLM	GPT-4o (OpenAI, 2024)	-	79.65	76.71
	Gemini2.5-Flash (Gemini Team, Google, 2025)	-	73.65	81.39
	Gemini2.5-Pro (Gemini Team, Google, 2025)	-	82.99	84.61
SLM (Qwen3-4B-Inst)	Naive	-	65.58	62.91
	Self-Refine (*turnwise*) (Madaan et al., 2023)	NeurIPS 2023	57.37	60.14
	Self-Refine (*planning*) (Madaan et al., 2023)	NeurIPS 2023	61.23	63.24
	Guide&Refine (*turnwise*) (Chen et al., 2025)	EMNLP 2025	73.31	72.14
	Guide&Refine (*planning*) (Chen et al., 2025)	EMNLP 2025	80.26	76.31
	InferAct (Fang et al., 2025)	EMNLP 2025	74.55	73.52
	SIPPER (Ours)		**86.30**	**85.39**

Table 2: **Comparative results of structured reasoning baselines and the proposed SIPPER for different models, including SLMs and proprietary LLMs.** Average accuracy (%) of trajectory and completion are reported on the MCPEval dataset.

degradation observed in Self-Refine. While Self-Refine (*planning*) achieves 63.24%, its *turnwise* version drops to 60.14%, falling below the naive SLM baseline of 62.91%. This indicates that the explicit intervention from SLM itself is counterproductive in complex agentic workflows. Rather than correcting errors, the SLM's persistent verification tends to corrupt the inherent logic.

Fragility of iterative LLM-aided feedback: Methods that incorporate external LLM intelligence, such as Guide&Refine (76.31% in *planning* ver.) and InferAct (73.52%), naturally outperform the SLM-only Self-Refine. However, they still exhibit a significant performance gap compared to **SIPPER**. These baselines rely on intricate feedback loops and few-shot expert corrections, which exacerbates the context-length bottleneck in SLMs. Given the computational overhead from the increased contexts and the complexity, the gains of these baselines are incremental.

Furthermore, Notably, Guide&Refine performs better when restricted to the initial planning stage (76.31%) compared to its turnwise execution (72.14%), confirming again our hypothesis that constructing a consistent, pre-defined structure throughout a complex, dynamic trajectory is inherently difficult for SLMs.

Superiority of SIPPER 's Structured Forcing: In contrast, **SIPPER** identifies and exploits the primacy of initial planning with precision. By anchoring the SLM in a streamlined, expert-guided reasoning template at the very onset, we establish a robust logical foundation that prevents stochastic drift.

3.7 Analysis

Here, we analyze the efficacy of **SIPPER**. Our analysis is based on the overarching research question:

Model		Naive	Sequentially-prefilled ST Reasoning		
			Initial Planning	Turnwise	Final Synthesis
Gemini2.5-Flash	Traj. (%)	73.65	75.58	73.81	74.22
	Comp. (%)	80.64	82.41	80.58	81.33
Qwen3-4B-Inst	Traj. (%)	65.58	86.30	66.29	66.41
	Comp. (%)	62.91	85.39	68.68	74.53

Table 3: **Importance of the initial planning in structured (ST) reasoning.** Average completion accuracy (%) is reported on agentic tasks from the MCPEval benchmark.

- How does sequentially-prefilled structured anchoring at the initial planning stage impact the procedural reliability and task success of SLM agents compared to autonomous reasoning or later-stage interventions?

RQ1: Why is the initial planning the critical bottleneck for SLMs? To investigate the efficacy of structured reasoning across different model scales, we compare the performance impact of our intervention on both LLMs and SLMs. We exclude frontier "thinking" models (e.g., Gemini2.5 Pro) in this analysis since their inherent reasoning traces preclude the external imposition of pre-defined structured reasoning. Instead, we utilize Gemini2.5 Flash as the LLM baseline in this analysis.

We consider three intervention points: *initial planning*, *turnwise*, and *final synthesis*. For the *turnwise* intervention, since reasoning requirements evolve dynamically, we curate three pre-defined sets of sequential prefills from a validation set. To isolate the impact of the reasoning structure itself, we employ Gemini2.5 Pro to select the most appropriate set based on the current trajectory at each turn. As shown in Table 3, Gemini2.5 Flash exhibits marginal performance gains across all intervention stages. This means that LLMs possess the capacity

Model	Metric	Intervention at Initial Planning	
		Static Guidance	**SIPPER** (Ours)
Qwen3-4B-Inst	Traj. (%)	65.67	86.30
	Comp. (%)	67.15	85.39

Table 4: **Effectiveness of dynamic structured reasoning over static guidance.** We compare providing expert-guided logic as a static prompt *vs* our sequentially-prefilled structured reasoning.

Method	Backbone	Performance		Avg. Tokens
		Traj. (%)	Comp. (%)	
Baseline (w/o CoT)	Qwen3-4B-Inst	65.58	62.91	-
Naive CoT	Qwen3-4B-Inst	66.23	67.27	0.5k
Autonomous Think	Qwen3-4B-Think	57.70	64.96	1.3k
SIPPER (Ours)	Qwen3-4B-Inst	86.30	85.39	0.3k

Table 5: **Comparison with autonomous reasoning.** **SIPPER** achieves higher procedural accuracy with significantly lower token overhead compared to the CoT and the thinking model, Qwen3-4B-Think.

to internalize instructions at any stages, though the improvements remain incremental as these models already possess substantial latent reasoning capabilities.

In contrast, SLMs exhibit a distinct sensitivity to the timing of intervention. Qwen3-4B-Inst shows a disproportionate performance surge specifically during the initial planning stage. The performance gap between *initial planning* and *turnwise* interventions shows that SLMs struggle to process complex, shifting instructions mid-trajectory. Furthermore, the *turnwise* approach is inherently reliant on an LLM for template selection; as detailed in Appendix, tasking the SLM with self-selecting the reasoning set results in severe performance degradation. Also, intervention at the *final synthesis* stage proves ineffective for SLMs, as early-stage logical errors typically result in a corrupted trajectory that lacks sufficient evidence for a correct conclusion. These results confirm that the initial planning is the most critical leverage point of structured reasoning for SLMs.

RQ2: Static Guidance *vs* Dynamic Planning (SIPPER): To isolate the impact of the prefill and generation mechanism in reasoning, we compare **SIPPER** against a static guidance. In this setup, a fixed, rigid block of high-level instructional text (refer to Fig. 4 in Appendix) is injected, immediately before the first tool invocation. Notice that, unlike structured reasoning, there is no rationale generation in response to the static guidance.

In Table 4, the results reveal that the limitation of the static guidance lies in its context-agnostic nature; because the guidance is static, it lacks the target-specific planning required to navigate the nuances of a particular user query. In contrast, **SIPPER** forces the SLM to dynamically generate rationales (g_k) between the prefilled anchors (e_k). This structure compels the model to map the universal expert logic onto the specific constraints of the current task.

RQ3: Can Structured Reasoning Outperform Autonomous Long-Thought? We compare **SIPPER** with autonomous reasoning, including naive chain-of-thought in Qwen3-4B-Inst and the long-thought model, Qwen3-4B-Think. From this, we evaluate whether the extensive test-time scaling inherent in autonomous reasoning translates into agentic success for SLMs.

Our results, summarized in Table 5, reveal a clear efficiency-quality paradox in autonomous SLM reasoning. While models like Qwen3-4B-Think generate significantly more reasoning tokens (in average, 1.3k), this verbosity does not yield proportional gains in accuracy. Instead, we observed frequent 'reasoning circularity'—where the model becomes trapped in self-referential logical loops—and 'hallucinatory drift', where the generated plan loses alignment with the available tool specifications.

In contrast, **SIPPER** achieves superior results, while generating much fewer tokens. By replacing unconstrained generation with expert-guided sequential prefilling, **SIPPER** effectively prunes the reasoning space, preventing the model from deviating into irrelevant logical paths. Furthermore, the integration of KV-cache injection ensures that this high-fidelity reasoning is delivered with minimal latency. This demonstrates that for on-device agentic workflows, structured anchoring is a more robust scaling strategy than purely autonomous test-time computation, providing the logical depth required for complex tasks without the prohibitive overhead of "long-thought" verbosity.

On-device Latency: To validate the practical usefulness of **SIPPER**, we compare it with the thinking model (Qwen3-4B-Think) in terms of latency which is measured on the Snapdragon 8 Elite Gen 5. Since the thinking model's reasoning needs heavy sequential decoding, the reasoning for initial planning needs 48.05s in average. Whereas, in **SIPPER**, the total reasoning token count is lower as in Table 5, and furthermore, the expert anchors are prefilled which incurs lower latency than decoding.

The initial planning latency in **SIPPER** is 5.48s. When we can use the pre-computed KV caches of the expert anchors, the latency is further reduced. Therefore, with the proposed reasoning anchoring, we can ensure responsive and deterministic performance on edge devices.

4 Related Works

4.1 LLM as Planner

Recent advancements have established LLMs as capable planners for task decomposition and sequential decision-making. LLM-Planner (Song et al., 2023) and KnowAgent (Zhu et al., 2025) demonstrate that LLMs can map natural language instructions into executable subgoals by leveraging few-shot prompting or external action knowledge bases. To better align with human intent, LLM-Personalize (Han et al., 2025) further incorporates iterative self-training and scene-graph structures. However, they primarily rely on the high-level reasoning of large-scale models, then may suffer from planning hallucinations or logical inconsistencies when scaled down to SLMs. To mitigate these inconsistencies, in (Wu et al., 2024; Chen et al., 2024b), topological structures are exploited to govern the planning process. Chen et al. (2024b) systematically explore multiple trajectory candidates via iterative branching and evaluation. Wu et al. (2024) utilize auxiliary neural modules to resolve complex task dependencies. Although these structural priors enhance planning robustness, they introduce significant computational overhead; tree search often incurs higher latency, and graph-based models deviate from the efficiency of a single-chain inference process.

While COPE (Lee et al., 2026) attempts to address efficiency via SLM-LLM collaboration, it still necessitates a "cascade" structure where the large model intervenes when the SLM fails, indicating a lack of standalone robustness in smaller models. In contrast, our structured reasoning method, **SIPPER**, enables robust single-chain planning without the prohibitive overhead of topological search or the structural complexity of auxiliary modules.

4.2 Structured Reasoning

Current research in structured reasoning can be divided into learning-based optimization and inference-time prompting strategies. Learning-based methods integrate structural priors directly into the model's parameters: SAND (Xia et al., 2025) finetunes agents through execution-guided critique of action deliberations, SEER (Chen et al., 2024a) optimizes hierarchical dependencies via reinforcement learning with structure-based returns, and Yubo Dong (2026) utilizes a graph-based framework with Max-Flow rewards to improve reasoning efficiency. While these methods achieve high fidelity, they entail significant training complexity. Conversely, non-learning approaches utilize the inherent capabilities of LLMs without parameter updates: RCI (Kim et al., 2023) and Self-Refine (Madaan et al., 2023) employ recursive feedback loops for iterative improvement, while InferAct (Fang et al., 2025) leverages multiple additional models to preemptively detect misaligned actions. To stabilize multi-step inference, Guide&Refine (Chen et al., 2025) extracts structured guidelines from successful trajectories, and IRCoT (Trivedi et al., 2023) interleaves retrieval with reasoning steps to maintain factual consistency.

In spite of their efficacy in LLMs, they have not considered the SLMs' properties. Learning-based frameworks require prohibitive computational resources for reinforcement learning, while prompting-based refinement introduces excessive latency and relies on the correction capabilities of frontier LLMs, which are notably deficient in smaller-scale models.

5 Conclusions

In this work, we introduced **SIPPER**, a framework designed to unlock the reliability of SLM-based agents through the primacy of initial planning. Our investigation reveals that the failure of SLMs in complex orchestration is not due to a lack of tool-use capability, but rather the absence of a robust logical foundation at the onset of a task. By employing structured logical forcing via expert-guided sequential prefilling, **SIPPER** ensures reasoning consistency and prevents the stochastic drift typical of autonomous SLMs. Furthermore, our approach can maximize on-device efficiency by leveraging system-level KV-cache re-use, achieving performance levels comparable to frontier models like GPT-4o without the overhead of massive test-time scaling. These results demonstrate that strategic, structured intervention at the planning stage is a more effective and scalable path for deploying sophisticated agentic systems on edge devices than increasing model parameter count or purely autonomous reasoning.

6 Limitations

Despite the performance gains demonstrated by **SIPPER**, several limitations remain to be addressed in future work. First, while our topic-agnostic reasoning templates were distilled from frontier models to ensure broad applicability, they may not capture the optimal reasoning path for highly specialized or niche domains. In such domains, we may need more complex prompt optimization techniques. Second, **SIPPER** currently focuses on single-agent planning and execution. In complex environments requiring multi-agent coordination or hierarchical orchestration, the structured prefilling logic might need to be extended to account for inter-agent communication protocols. Its application to collaborative multi-agent systems remains an open area for investigation. Finally, although our KV-cache injection mechanism significantly reduces inference latency, the initial prefilling of anchor tokens still incurs a marginal computational cost during the first forward pass. While this cost is much lower than other approaches or thinking models, and also negligible compared to the total execution time of multi-turn agentic workflows, further optimization of the prefilling process could enhance responsiveness for real-time edge applications.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *ArXiv:2303.08774*.

Baosong Yang Beichen Zhang Binyuan Hui Bo Zheng Bowen Yu Chang Gao Chengen Huang Chenxu Lv Chujie Zheng Dayiheng Liu Fan Zhou Fei Huang et al. An Yang, Anfeng Li. 2023. Qwen3 technical report. *Arxiv:2505.09388*.

Jihwan Bang, Juntae Lee, Seunghan Yang, and Sungha Choi. 2025. Think straight, stop smart: Structured reasoning for efficient multi-hop rag. In *Conference on Neural Information Processing Systems (workshop)*.

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. *https://arxiv.org/abs/2506.02153*.

Guoxin Chen, Kexin Tang, Chao Yang, Fuying Ye, Yu Qiao, and Yiming Qian. 2024a. SEER: Facilitating structured reasoning and explanation via reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Jiaxiang Chen, Zhuo Wang, Mingxi Zou, Zhucong Li, Zhijian Zhou, Song Wang, and Zenglin Xu. 2025. From implicit exploration to structured reasoning: Leveraging guideline and refinement for llms. In *Findings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. 2024b. When is tree search useful for LLM planning? it depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Claude Code. 2026. Claude code. https://claude.com/product/claude-code.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *ArXiv:2407.21783*.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2025. Preemptive detection and correction of misaligned actions in LLM agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Gemini Team, Google. 2025. Gemini 2.5 pro: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. https://arxiv.org/abs/2507.06261. Technical Report.

Dongge Han, Trevor McInroe, Adam Jelley, Stefano V. Albrecht, Peter Bell, and Amos Storkey. 2025. LLM-personalize: Aligning LLM planners with human preferences via reinforced self-training for housekeeping robots. In *Proceedings of the 31st International Conference on Computational Linguistics*.

HuggingFace. 2026. Huggingface. https://huggingface.co/.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Conference on Neural Information Processing Systems*.

Byeongchan Lee, Jonghoon Lee, Dongyoung Kim, Jaehyung Kim, Kyungjoon Park, Dongjun Lee, and Jinwoo Shin. 2026. Efficient llm collaboration via planning. *https://arxiv.org/abs/2506.11578*.

Zhiwei Liu, Jielin Qiu, Shiyu Wang, Jianguo Zhang, Zuxin Liu, Roshan Ram, Haolin Chen, Weiran Yao, Shelby Heinecke, Silvio Savarese, Huan Wang, and Caiming Xiong. 2025. MCPEval: Automatic MCP-based deep evaluation for AI agent models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Conference on Neural Information Processing Systems*.

Manus AI. 2026. Manus ai. https://https://manus.im/.

OpenAI. 2024. Gpt-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2024-05-13.

OpenClaw. 2026. Openclaw. https://openclaw.ai/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representation*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *International Conference on Computer Vision*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *https://arxiv.org/abs/2201.11903*.

Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, and Dongsheng Li. 2024. Can graph learning improve planning in llm-based agents? In *Conference on Neural Information Processing Systems*.

Yu Xia, Yiran Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. 2025. Sand: Boosting llm agents with self-taught action deliberation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2025. Tree of Thoughts: Deliberate problem solving with large language models. In *Conference on Neural Information Processing Systems*.

Linchao Zhu Yi Yang Yubo Dong, Hehe Fan. 2026. Structured reasoning for llms: A unified framework for efficiency and explainability. In *International Conference on Learning Representation*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2025. KnowAgent: Knowledge-augmented planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

- Appendix -

Unlocking SLM Agents: Primacy of Initial Planning and Structured Sequentially-Prefilled Efficient Reasoning

A Implementation Details

All methods were implemented using the PyTorch framework and the Hugging Face (HuggingFace, 2026) Transformers library. Experiments were conducted on NVIDIA A100 (80GB) GPUs. For distributed processing and memory optimization, we utilized the accelerate library where applicable. To ensure the reproducibility of our results and minimize stochastic variance, we employed greedy decoding by setting the temperature to $T = 0$ for all generation tasks. Other decoding parameters (e.g., top_p, top_k) were disabled to maintain deterministic outputs, allowing for a rigorous comparison of model performance across different benchmarks.

Dataset Size: For the benchmark, we generated 100 tasks per topic to ensure a balanced and statistically significant distribution across diverse domains. The specific prompt templates and task-generation pipelines strictly adhere to the standards described in (Liu et al., 2025)

Implementation of Structured Reasoning: We evaluated two distinct prompting strategies: providing explicit instructions for reasoning within the user-side prompt versus prefilling the assistant's response with an initial reasoning header (e.g., a "thought" block). Experimentally, the former approach proved more effective. This is likely because modern instruction-tuned models are primarily optimized to satisfy constraints and stylistic requirements defined within the user context; by contrast, forcing a specific starting sequence in the assistant's turn can occasionally lead to a distribution shift that disrupts the model's natural generation flow or weakens its adherence to the requested logical structure. Consequently, we adopted user-directed instructions to trigger structured reasoning across our experiments.

B Results on More SLMs

To evaluate the generalizability of both our proposed framework and the structured reasoning baselines, we extended our experiments to a broader range of models, specifically Qwen3-1.7B and Qwen3-8B. As illustrated in Table 6, models without instruction tuning consistently exhibit inferior performance in agentic tasks compared to Qwen3-4B-Inst (86.30% in trajectory, 85.39% in completion). Notably, even the larger Qwen3-8B variant fails to surpass the 4B instructed model when alignment is absent—a trend that persists across all evaluated baselines. This performance gap highlights the inherent difficulty non-tuned models face in processing complex reasoning instructions. In this challenging setting, baseline methods suffer from severe performance degradation, whereas our proposed method, **SIPPER**, demonstrates superior robustness by leveraging a streamlined single reasoning chain. Furthermore, across all models, the comparative analysis of reasoning strategies (naive, Autonomous thinking, turnwise, and final synthesis) shows similar trends observed with Qwen3-4B-Inst. These consistent empirical results support our central contribution: the critical importance of expert intervention during the initial planning stage for successful task execution.

Method	Traj. (%)	Comp. (%)
Naive (Qwen3-1.7B)	34.94	26.66
Autonomous Think (thinking mode)	22.45	13.18
Self-Refine (*turnwise*) (Madaan et al., 2023)	19.23	13.51
Self-Refine (*planning*) (Madaan et al., 2023)	20.12	17.22
Guide&Refine (*turnwise*) (Chen et al., 2025)	28.82	22.21
Guide&Refine (*planning*) (Chen et al., 2025)	27.35	24.32
InferAct (Fang et al., 2025)	31.42	23.89
turnwise	24.47	20.58
final synthesis	32.64	15.84
SIPPER (Ours)	**42.14**	**29.31**
Method	Traj. (%)	Comp. (%)
Naive (Qwen3-8B)	52.60	60.62
Autonomous Think (thinking mode)	58.48	63.21
Self-Refine (*turnwise*) (Madaan et al., 2023)	50.71	58.34
Self-Refine (*planning*) (Madaan et al., 2023)	52.04	61.22
Guide&Refine (*turnwise*) (Chen et al., 2025)	58.08	71.22
Guide&Refine (*planning*) (Chen et al., 2025)	62.02	73.14
InferAct (Fang et al., 2025)	59.43	70.49
turnwise	47.45	68.32
final synthesis	50.29	64.13
SIPPER (Ours)	**62.31**	**78.86**

Table 6: **Comparative results of structured reasoning baselines and variants of SIPPER, the proposed SIPPER for two SLM models (Qwen3-1.7B (top) and Qwen3-8B (bottem)).** Average accuracy (%) of trajectory and completion are reported on the MCPEval dataset.

C Prompt Template for SIPPER

Prompt template for SIPPER

```
You are currently in **Phase 1: Initial Planning**. Do NOT call any tools yet. Your goal is to analyze the request and create a
 solid plan through 3 Steps. Follow the structured steps provided in the prompt.\n\n In this turn, ONLY answer the Step1. Do
Not proceed to the Steps2 and 3 in this turn. **Step 1: Entity & Constraint Analysis**\n Identify the core entities and

specific constraints from the goal and title of the given task.\n\n- **Core Entities** (Who/What/Where):
```

```
In this turn, ONLY answer the step2. Do Not proceed to the next Steps in this turn. Namely, Only identify the necessary tools
and **pre-calculate** any specific parameters (Dates, IDs) required for the first steps. If logic (like adding days) is needed,
 show the calculation.
```

```
Now Step3. Based on this reasoning, ONLY outline the concise execution steps.
```

Figure 3: Prompt template for initial planning in **SIPPER**.

D Prompt Template for Static Guidance Variant

Prompt template for static guidance

```
Before generating the next tool call or final response, you MUST verify the following:

1.  **Goal Alignment**: Does my next action directly contribute to the user's original goal?
2.  **Contextual Consistency**: Am I using the correct identifiers (IDs, keys, names) from the PREVIOUS steps? Do not confuse
information from different entities.

3.  **Tool Schema Adherence**: If calling a tool, does the tool name and all its parameters EXACTLY match the provided schema?
Are all required parameters present and correctly formatted?

4.  **Feasibility Check**: Is the task still possible with the available information? If a previous tool call returned empty or
 insufficient data, I must report this limitation instead of guessing or calling another tool with invalid inputs.
```

Figure 4: Prompt template for the static guidance in Table 4.

E Prompt Template for *turnwise* Variant in SIPPER

Type1: Prompt template for plan

```
To tackle this task, first clarify the user's ultimate goal in one sentence.
```

```
Good. Now, list the specific pieces of information you need to gather to achieve this goal, less than 300 tokens.
```

```
Based on that, what is the immediate first sub-task you need to perform? Describe it briefly.
```

```
Which tool is best suited for this sub-task and why? Describe it briefly.
```

Type2: Prompt template for checking tool argument

```
Which tool have you decided to call? Describe it briefly.
```

```
Check the tool definition. What are the specific required parameters for this tool? Describe it briefly.
```

```
Based on the conversation history and context, determine the exact values for these arguments. Describe it briefly.
```

Type3: Prompt template for observation success

```
Analyze the tool output. What key data or facts have you successfully gathered? Describe it briefly.
```

```
Verify this data against the user's request. Does it confirm any specific facts?  Describe it briefly.
```

```
With this information in hand, what is the next logical step to move forward?  Describe it briefly.
```

Type4: Prompt template for observation error

```
The tool execution failed or returned no results. Analyze the potential root cause (e.g., typo, invalid ID, filter too strict),
 briefly.
```

```
Do not give up. Propose a concrete alternative strategy or fallback plan to get the information, less than 300 tokens.
```

```
Specify the new search query or parameter adjustment you will use for this attempt.
```

Type5: Prompt template for repeated same observation error

```
This tool execution failed or returned no results, repeatedly. Check the potential root cause is tool itself and consider using
 other tools. What is the next logical step to move forward?
```

Type6: Prompt template for filtering

```
Review the retrieved list. What specific condition or criteria must the selected item meet according to the user? Describe it
briefly.
```

```
Iterate through the candidate items. Analyze the first few items to see if they match the criteria, briefly.
```

```
Identify the final selected item that best matches the criteria.
```

Type7: Prompt template for final response synthesis

```
The task is nearly complete. Summarize the key findings you have gathered so far, point by point. Do this less than 500 tokens.
```

```
Synthesize these findings into a final conclusion. Do this less than 500 tokens.
```

```
Draft the final response to the user. What key details or caveats must be included? Do this less than 500 tokens.
```

Prompt template for type selection via LLM

```
# Context

- **User Goal**: {task.goal}
- **Available Tools**: {tools_list_str}


# Conversation History (Last few turns)
{history_str}


# Candidate Templates
1. **TEMPLATE_PLAN**: Use when starting a new task or replanning. (User input -> Plan)
2. **TEMPLATE_TOOL_ARGS**: Use when a specific tool needs to be called, to reason about parameters.

3. **TEMPLATE_OBSERVATION_SUCCESS**: Use when a tool returned valid data. Analyze and move to next step.
4. **TEMPLATE_OBSERVATION_ERROR**: Use when a tool returned an error or empty result. Plan a fallback.

5. **TEMPLATE_OBSERVATION_REPEATED_SAME_ERROR**: Use when a tool returned repeated (more than 2 times) error or empty result.
Plan to use another tool.
6. **TEMPLATE_FILTERING**: Use when a list of items was returned and needs filtering by condition.

7. **TEMPLATE_SYNTHESIS**: Use when all info is gathered and a final answer is needed.

# Decision Logic

- If last message is **User** -> **TEMPLATE_PLAN**.
- If last message is **Tool Output**:
    - Error/Empty -> **TEMPLATE_OBSERVATION_ERROR**

    - Repeated Error/Empty in a specific tool: **TEMPLATE_OBSERVATION_REPEATED_SAME_ERROR**
    - List needing selection -> **TEMPLATE_FILTERING**
    - Good data -> **TEMPLATE_OBSERVATION_SUCCESS** or **TEMPLATE_SYNTHESIS** (if done).

- If Agent intended to use a tool but hasn't called it -> **TEMPLATE_TOOL_ARGS**.

# Output Format

Return ONLY the JSON object: {{"selected_template_id": "TEMPLATE_NAME"}}
```

Figure 5: Prompt template for *turnwise* in Table 3.