SVP: STATE-VERIFIED PLANNING FOR SYSTEM-LEVEL EXTERNAL KNOWLEDGE INTEGRATION IN AGENTIC REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Grounding agentic LLMs in external knowledge can reduce hallucinations and improve reliability in knowledge-intensive and domain-specific reasoning tasks. However, effective integration requires more than providing access to search or tools. In complex multi-step settings, the acquired evidence may challenge assumptions or reshape reasoning, while the unsupported conclusions can propagate when cross-step progress is not explicitly maintained. We propose State-Verified Planning (SVP), a system-level framework that coordinates external knowledge through explicit cross-step state and adaptive control. SVP decomposes a query into a revisable high-level plan, executes each subgoal with a step-local ReAct-style executor, and maintains a compact State Box containing the plan, completed-step conclusions, available tools, and tool budget. After each step, a state maintainer records compact progress and derives a structured step view for assessment, allowing subsequent execution to condition on accumulated outcomes while supporting selective evidence refinement and plan revision when needed. Across three knowledge-intensive reasoning benchmarks, SVP consistently outperforms direct answering, module-level tool-augmented methods, and a system-level agentic baseline. On HLE-Verified Gold, SVP achieves 51.20% accuracy, outperforming the strongest baseline by 2.10 points, and improves further with larger tool budgets. The cross-model evaluation, component analysis, and a case study further demonstrate its robustness and the value of state-mediated external knowledge coordination across step-local reasoning and system-level orchestration.

1 INTRODUCTION

Large Language Models (LLMs) have achieved strong performance on complex reasoning tasks. However, their reliability remains limited in knowledge-intensive and domain-specific settings, where their parametric knowledge may be incomplete, outdated, or insufficient for the task Li et al. (2025a). To address this limitation, external retrieval and tools ground inference in task-specific evidence, helping mitigate knowledge-related hallucinations Rahman et al. (2026).

Recent approaches to external knowledge integration have expanded beyond the retrieve-then-generate paradigm of RAG. Agentic methods enable LLMs to formulate and refine queries, invoke search during reasoning, integrate retrieved evidence, and continue information seeking across turns Yao et al. (2022); Deng et al. (2026). More broadly, complex applications have motivated agentic workflows that coordinate specialized components or agents for planning, execution, memory, and verification Chowa et al. (2026). Despite these advances, acquired knowledge is often used within the active reasoning trajectory, with limited explicit support for carrying its implications into later execution and plan revision.

This limitation is consequential in multi-step reasoning, where evidence acquired for one subgoal may invalidate assumptions in the remaining plan or alter how later subgoals should be executed. When intermediate outcomes remain in growing reasoning traces, early errors can propagate across subsequent steps Xinjie et al. (2025). Moreover, static plans may preserve unsupported assumptions even when acquired evidence calls them into question You et al. (2026). Together, these challenges

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

motivate a system-level approach that maintains an explicit representation of cross-step progress and allows external knowledge to inform both step-local reasoning and subsequent task orchestration.

To this end, we propose *State-Verified Planning (SVP)*, a state-mediated framework for system-level external knowledge integration in agentic reasoning. SVP begins with a revisable high-level plan and executes each subgoal through a step-local ReAct process. Rather than propagating complete reasoning traces across steps, it maintains a compact and evolving *State Box* containing the current plan, completed-step conclusions, and execution metadata, providing an explicit cross-step state for subsequent execution and orchestration. Retrieved content is processed by an extractor before use. After each step, a state maintainer records a compact conclusion for subsequent state construction and derives a structured step view from current claims, unresolved questions, and recent reasoning. The critic assesses this view with the State Box and accumulated evidence, seeks additional evidence when necessary, and decides whether to continue or trigger plan revision. This state-mediated loop allows external knowledge to inform both step-local reasoning and subsequent task orchestration.

Our empirical evaluation focuses on knowledge-intensive question answering across benchmarks with diverse reasoning challenges, using web search as the external knowledge interface. We evaluate SVP on HLE-Verified Gold, GPQA-Diamond, and SimpleQA-Verified-300, achieving the best results among the evaluated reasoning, tool-augmented, and agentic methods. Tool-budget analysis shows that SVP translates additional tool capacity into consistent accuracy gains, unlike ReAct, while its advantage is not explained solely by greater token consumption. Cross-model evaluation further shows consistent gains across additional backbone LLMs. Component analysis examines key framework design choices, while a representative case study illustrates state-mediated trajectory correction. Our main contributions are as follows:

- We develop SVP, a system-level framework that utilizes the explicit cross-step state to coordinate step-local ReAct execution with subsequent task orchestration, supporting the use of externally acquired knowledge across subsequent reasoning and orchestration.
- We introduce a dynamically updated State Box that maintains a compact representation of the current plan, completed-step conclusions, and execution metadata, serving as an explicit control interface between step-local execution and system-level orchestration.
- We demonstrate consistent improvements across multiple benchmarks and conduct extensive evaluations covering the tool-budget scaling, cross-model effectiveness, component analysis, and reasoning behavior.

2 Related Work

Tool-Augmented Reasoning. LLMs remain constrained by incomplete or outdated parametric knowledge, motivating the use of retrieval and external tools. Retrieval-augmented generation conditions model outputs on retrieved passages Lewis et al. (2020), WebGPT enables models to search and cite online sources Nakano et al. (2021), and Toolformer teaches LLMs to invoke appropriate external APIs Schick et al. (2023). While effective for directly retrievable information, complex tasks often require reasoning to be interleaved with active knowledge acquisition. ReAct establishes this interaction by interleaving reasoning traces, tool actions, and observations Yao et al. (2022). Subsequent methods improve different aspects of this process: Self-Ask and FLARE employ query decomposition and dynamic retrieval Press et al. (2023); Jiang et al. (2023), while CRITIC and Self-RAG use tool feedback or reflection to assess generated content Gou et al. (2024); Asai et al. (2024). More recently, Search-o1 integrates agentic search into the reasoning process and introduces a Reason-in-Documents module to extract relevant information from retrieved documents before returning it to the reasoning trajectory Li et al. (2025b). Training-based approaches such as UR^2 and HiPRAG further use reinforcement learning to coordinate retrieval and reasoning and reduce inefficient search behavior Li et al. (2026); Wu et al. (2025b). These studies improve how external knowledge is acquired, processed, and evaluated, but often incorporate it within the active reasoning trajectory or a retrieval-specific module. SVP instead frames external knowledge integration as a system-level coordination problem, using explicit cross-step state to carry forward compact reasoning outcomes and execution context for subsequent reasoning and task orchestration.

Agentic Orchestration. Beyond local interactions between reasoning and action, complex tasks have motivated workflow-level orchestration of planning, execution, memory, and feedback. Plan-

and-Solve separates high-level planning from stepwise execution Wang et al. (2023), while Tree of Thoughts enables the exploration and evaluation of alternative reasoning paths Yao et al. (2023). LATS further combines reasoning, acting, planning, and environmental feedback through tree search Zhou et al. (2023), and Reflexion uses verbal feedback stored in episodic memory to improve subsequent trials Shinn et al. (2023). Together, these methods illustrate how explicit planning, persistent information, and intermediate feedback can support reasoning beyond a single uninterrupted trajectory. Recent systems extend this perspective to broader agentic workflows. WebThinker interleaves reasoning, autonomous web exploration, and report drafting for deep-research tasks Li et al. (2025c), while Agentic Reasoning coordinates web search, code execution, and structured Mind-Map memory for knowledge-intensive reasoning Wu et al. (2025a). Verification-oriented frameworks further incorporate intermediate evaluation into multi-agent orchestration. VeriMAP uses planner-defined verification functions to validate dependent subtasks and improve coordination among collaborating agents Xu et al. (2026), while VMAO organizes specialized agents in a directed acyclic graph and follows a cycle of planning, execution, verification, and replanning for complex query resolution Zhang et al. (2026). These studies demonstrate the value of coordinating planning, persistent information, and verification across workflows. Building on this orchestration perspective, SVP uses a dynamically updated State Box as an explicit cross-step interface linking step-local execution with system-level orchestration.

3 METHODOLOGY

We formulate SVP by extending ReAct-style knowledge interaction from a single reasoning trajectory to a state-mediated closed-loop process. Given a query q, external tools $\mathcal{T}$, and the current State Box S_t, each iteration is summarized as $(\tau_t, Z_t, a_t, S_{t+1}) = \mathcal{F}_{\mathrm{SVP}}(q, S_t; \mathcal{T})$, where τ_t denotes step-local reasoning, Z_t is a structured view of the current step, a_t is a critic-guided control action, and S_{t+1} is the State Box for subsequent execution. This formulation supports system-level external knowledge integration across step-local reasoning and subsequent task orchestration.

3.1 OVERVIEW OF THE SVP FRAMEWORK

As illustrated in Figure 1, SVP organizes external knowledge integration as a state-mediated closed loop of planning, step-local execution, state maintenance, and critic-guided control. The planner creates a revisable high-level plan. At each step, the ReAct executor reasons from the original task input and the current State Box, with retrieved web content processed by the extractor before use. The state maintainer then derives a structured step view for critic assessment and records a compact conclusion for subsequent state construction. Using the current State Box, structured step view, and accumulated evidence, the critic assesses progress, seeks additional evidence when necessary, and either advances execution or returns control to the planner for replanning. The resulting action determines the State Box used for subsequent execution. Once the loop completes all steps, the final answer is synthesized from the finalized reasoning context.

3.2 HIGH-LEVEL PLANNING

Complex multi-step reasoning requires global structure, but a fixed plan may preserve assumptions that later evidence challenges. SVP therefore maintains a revisable high-level plan that organizes the task while allowing subsequent steps to adapt to evidence during execution. Given an input q, the planner Π generates an ordered sequence of subgoals:

$$P^{(0)} = \Pi(q) = (g_1, g_2, \ldots, g_n)\,, \tag{1}$$

where g_i denotes the objective of the i-th step. The plan is stored in the State Box, and the step prompt directs the executor to the current subgoal. When the critic requests replanning after step t, the planner receives the original input and the planning and reasoning history H_t. If $P^{(k)}$ denotes the current plan, the planner retains the completed prefix and regenerates only the remaining subgoals:

$$P^{(k+1)} = P^{(k)}_{\leq t} \oplus \Pi_{\mathrm{rem}}(q, H_t), \tag{2}$$

where $P^{(k)}_{\leq t}$ is the completed prefix of the current plan, Π_{rem} generates the revised remaining steps, and $\oplus$ denotes sequence concatenation. This allows evidence acquired during execution to reshape subsequent task orchestration without discarding completed progress.

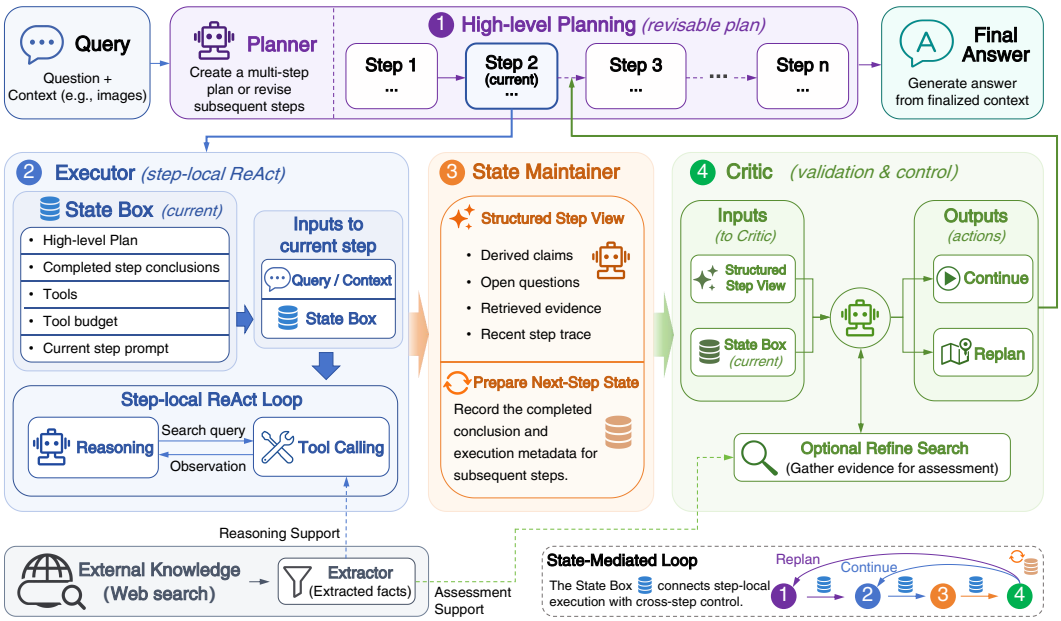

Figure 1: Overview of State-Verified Planning (SVP). The planner creates or revises a multi-step plan. At each step, the executor performs step-local ReAct reasoning from the query context and current State Box. The state maintainer records a compact conclusion and derives a structured step view for critic assessment. The critic may request additional evidence before deciding whether to continue execution or trigger replanning. Retrieved content is processed by the extractor before supporting executor-side reasoning or critic-side assessment. The control action determines the next State Box. Once all steps are completed, the final answer is generated from the finalized context.

3.3 State-Guided Execution

Executing each step with an accumulated reasoning history can obscure relevant progress and introduce unnecessary context. SVP therefore combines step-local execution with a compact State Box. At step t, the State Box is defined as

$$S_t = \left(P^{(k)}, C_{<t}, \mathcal{T}, b_t, \rho_t\right), \tag{3}$$

where $P^{(k)}$ is the current plan version, $C_{<t}$ contains the conclusions of completed steps, $\mathcal{T}$ denotes the available tools, b_t is the remaining tool-call budget, and ρ_t is the current step prompt. The State Box serves as a task-local control state rather than a general-purpose memory store. It follows a fixed schema, is updated after each step, and retains only the information required for subsequent execution and control. The executor $\mathcal{E}$ receives only the original input and the current State Box:

$$\tau_t = \mathcal{E}(q, S_t), \tag{4}$$

where τ_t denotes the step-local reasoning trajectory. Full reasoning traces from previous steps are omitted, while relevant outcomes are retained in $C_{<t}$. Within each step, the executor interleaves reasoning and tool calls in a ReAct-style process, subject to the budget b_t. For the j-th search action $u_{t,j}$, the retrieved content is processed by the extractor $\mathcal{X}$:

$$R_{t,j} = \text{Search}(u_{t,j}), \qquad e_{t,j} = \mathcal{X}(u_{t,j}, R_{t,j}), \tag{5}$$

where $R_{t,j}$ is the raw search result and $e_{t,j}$ is the extracted information returned to the executor for subsequent reasoning. After completing the subgoal, the executor passes its local trajectory to the state maintainer. This design supports external knowledge use within each step, while compact completed-step outcomes are carried forward through the shared State Box.

3.4 State Maintenance

Full step-local trajectories contain useful details but are cumbersome to propagate and difficult to assess consistently across steps. SVP therefore uses the state maintainer $\mathcal{M}$ to produce a compact

conclusion c_t for cross-step persistence and a structured step view Z_t for critic assessment. Evidence acquired during the current execution is first appended to the cumulative evidence state:

$$E_{\leq t} = E_{<t} \oplus E_t, \qquad \bar{E}_t = \mathcal{W}(E_{\leq t}), \tag{6}$$

where $\mathcal{W}$ retains a bounded recent evidence window. The state maintainer then produces

$$(c_t, Z_t) = \mathcal{M}(S_t, \tau_t, \bar{E}_t), \tag{7}$$

and updates the completed-step conclusions as

$$C_{\leq t} = C_{<t} \oplus (c_t). \tag{8}$$

Z_t contains current-step claims, unresolved questions, and a bounded excerpt of the trajectory. The evidence state remains separate from the State Box, while the compact conclusion c_t carries current-step outcomes into subsequent executor states. After the critic produces the action a_t, the next State Box is constructed as

$$S_{t+1} = \mathcal{U}(S_t, c_t, a_t), \tag{9}$$

where $\mathcal{U}$ incorporates c_t, the remaining tool budget, the resulting plan, and the next-step prompt. CONTINUE preserves the plan and advances execution, whereas REPLAN retains the completed prefix and revises only the remaining subgoals.

3.5 CRITIC-GUIDED CONTROL

An explicit state records cross-step progress, but does not by itself determine whether execution should proceed or the remaining plan should be revised. SVP therefore uses the critic $\mathcal{C}$ to assess the current step from the State Box S_t, structured step view Z_t, and evidence window $\bar{E}_t$. Together, these inputs allow the critic to assess support for the subgoal and its consistency with the remaining plan. When further evidence is needed, the critic issues an internal REFINE SEARCH request. The system executes the requested search using $\mathcal{T}$, processes the results through the extractor $\mathcal{X}$, and appends the extracted evidence to the accumulated evidence. The critic then reassesses the same step with an updated $\bar{E}_t$. These calls share the global tool budget and remain within the current step. After assessment and any internal refinement, the critic produces the cross-step action

$$a_t = \mathcal{C}(S_t, Z_t, \bar{E}_t; \mathcal{T}) \in \{\text{continue}, \text{replan}\}. \tag{10}$$

Both actions retain c_t. CONTINUE preserves the current plan and advances execution, whereas REPLAN preserves the completed prefix and revises only the remaining subgoals. This allows accumulated external evidence to affect both current-step validation and subsequent task orchestration.

3.6 ANSWER SYNTHESIS

Step-local execution isolates subgoals, while answer synthesis requires their completed reasoning to be integrated into a coherent response. After the control loop terminates, the answer synthesizer $\mathcal{A}$ generates the final response from the original query q and the finalized reasoning context Γ_T:

$$y = \mathcal{A}(q, \Gamma_T), \tag{11}$$

where y is the final answer, T is the number of completed steps, and Γ_T consists of the ordered State Box and step-local trajectory pairs $\{(S_t, \tau_t)\}_{t=1}^{T}$. These step-level contexts are assembled for answer synthesis and are not propagated between executor steps. If the accumulated context exceeds the model's context window, the earliest complete step blocks are removed until it fits. The retained context begins with S_i, whose $C_{<i}$ preserves compact conclusions from removed steps, while later retained blocks preserve detailed recent reasoning.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. *(1) HLE-Verified Gold (668).* Humanity's Last Exam (HLE) is a highly challenging benchmark for expert-level reasoning across diverse academic domains. We use the verified 668-item Gold subset released in HLE-Verified Zhai et al. (2026). *(2) GPQA-Diamond.* GPQA-Diamond Rein et al. (2023) comprises 198 challenging graduate-level multiple-choice questions in biology, physics, and chemistry. *(3) SimpleQA-Verified-300.* SimpleQA Verified Haas et al. (2025) evaluates short-form factuality. To keep agentic evaluation tractable, we use the first 300 examples in release order as a fixed, reproducible subset, with reference URLs withheld from all methods.

Baselines. We group baselines by how they incorporate external knowledge. *(1) No External Knowledge:* DirectQA answers directly, CoT Wei et al. (2022) performs step-by-step reasoning, and Plan + CoT adds high-level planning before reasoning. *(2) Module-Level Knowledge Integration:* ReAct Yao et al. (2022) interleaves reasoning and tool use, Plan + ReAct follows a fixed plan with ReAct, CRITIC Gou et al. (2024) performs tool-assisted critique and correction, and Search-o1 Li et al. (2025b) integrates search into the reasoning process. *(3) System-Level Knowledge Integration:* Agentic Reasoning Wu et al. (2025a) coordinates planning, reasoning, and tools in a multi-stage workflow as the system-level baseline. For controlled comparison, all tool-augmented methods share the same web search tool. Agentic Reasoning's additional code tool is disabled.

Evaluation Metrics. Accuracy (ACC) measures whether the reference answer appears in the prediction. Exact Match (EM) measures agreement after normalization. F1 measures token-level overlap between predicted and reference answers. Higher values indicate better performance.

Implementation Details. The primary reasoning model, DeepSeek-V4-Flash, is accessed through SiliconFlow. We set the temperature of all SiliconFlow-hosted models to 0.1 to reduce decoding stochasticity and improve comparability across runs. The maximum generation length is set to 16,384 tokens for all reasoning methods, while all remaining parameters retain their default values unless otherwise specified. Tool-augmented methods use Serper for Google-based web search and Jina Reader for webpage content extraction, with up to ten results returned per search. Except in the tool-budget analysis, all tool-augmented methods share a five-call search budget per question, which is explicitly stated in their prompts. For cross-model evaluation, Qwen3.5-9B and Gemma4-12B-IT are deployed with vLLM on NVIDIA H100 GPUs and use their model-recommended decoding settings. For image-based questions, Kimi K2.6, also accessed through SiliconFlow, provides the same textual description to all methods. Further implementation, inference, dataset, and evaluation details are provided in Appendix C and Appendix D.

4.2 Main Results

Table 1 presents the main results across the three benchmarks. Notably, the strongest competing method varies across benchmarks: Search-o1 on HLE-Verified Gold, CoT on GPQA-Diamond, and ReAct on SimpleQA-Verified-300, whereas SVP ranks first across all nine dataset-metric combinations. This highlights its cross-benchmark consistency under varying reasoning and external knowledge requirements. Detailed benchmark results including average search calls and token consumption are reported in Appendix E.1.

The comparison between DirectQA and CoT highlights the importance of reasoning. Compared with DirectQA, which disables Flash's thinking mode, CoT enables it and prompts step-by-step reasoning, improving ACC from 29.94% to 47.75% on HLE, from 69.19% to 84.85% on GPQA, and from 16.67% to 21.33% on SimpleQA. However, static planning provides no consistent benefit: Plan + CoT underperforms CoT, and Plan + ReAct underperforms ReAct, across three datasets. This suggests that an upfront plan alone is insufficient for robust coordination during execution.

On HLE-Verified Gold, ReAct and Search-o1 improve upon CoT, showing that external evidence can complement reasoning on challenging knowledge-intensive problems. SVP further raises ACC to 51.20%, outperforming the strongest baseline by 2.10 points. This result is consistent with the benefit of coordinating external knowledge across steps rather than only within the active reasoning trajectory. Agentic Reasoning is evaluated here under the standardized web-only setting without its original code tool. Its lower performance therefore reflects this controlled variant rather than the complete original tool suite.

On GPQA-Diamond, CoT obtains 84.85% ACC, indicating strong performance by DeepSeek-V4-Flash and limited room for improvement. In this strong-reasoning regime, most tool-augmented baselines do not surpass CoT, suggesting that retrieval offers limited benefit when the model solves many questions internally. Nevertheless, SVP achieves the highest ACC of 85.35%, surpassing CoT by a modest 0.50 points and remaining the only tool-augmented method above CoT.

On SimpleQA-Verified-300, methods without external knowledge achieve at most 21.33% ACC. With the reference URLs withheld, ReAct substantially improves ACC to 54.33% by autonomously acquiring relevant information. SVP further reaches 55.00% ACC and improves F1 from ReAct's

Table 1: Results on three knowledge-intensive benchmarks using DeepSeek-V4-Flash. The higher is better. The best and second-best results are indicated in **bold** and underlined, respectively.

	HLE-Verified Gold (668)			GPQA-Diamond			SimpleQA-Verified-300		
Method	ACC	EM	F1	ACC	EM	F1	ACC	EM	F1
No External Knowledge									
DirectQA (no thinking)	29.94	28.59	31.44	69.19	69.19	69.19	16.67	15.67	25.47
CoT	47.75	45.66	48.73	84.85	84.85	84.85	21.33	19.67	30.21
Plan + CoT	46.86	44.16	47.69	84.34	84.34	84.34	20.33	18.67	29.67
Module-Level Knowledge Integration									
ReAct	48.65	45.36	49.00	83.84	83.84	83.84	54.33	50.00	63.86
Plan + ReAct	46.71	43.71	46.97	81.31	81.31	81.31	53.00	46.00	62.94
CRITIC	46.86	42.22	45.34	80.81	80.30	80.30	22.67	19.00	31.51
Search-o1	49.10	45.81	49.17	78.79	78.79	78.79	36.00	31.33	45.69
System-Level Knowledge Integration									
Agentic Reasoning	45.51	42.37	45.32	80.81	80.81	80.81	15.00	13.00	21.31
SVP (Ours)	**51.20**	**48.65**	**52.68**	**85.35**	**85.35**	**85.35**	**55.00**	**50.67**	**67.73**

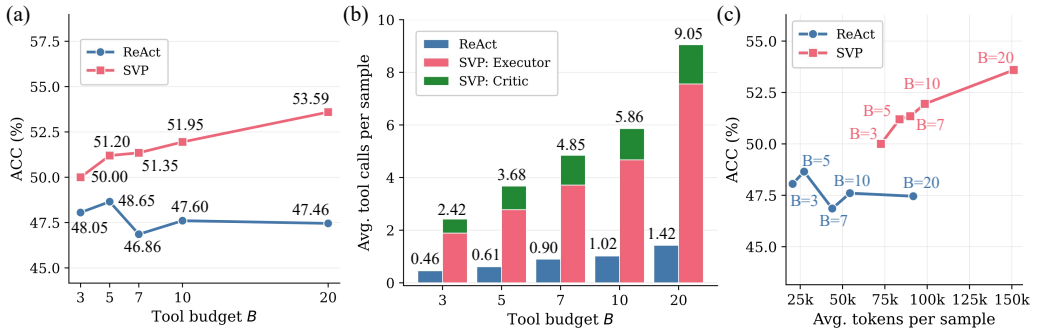

Figure 2: Scaling behavior across tool budgets on HLE-Verified Gold (668). (a) ACC as the tool budget increases. (b) Average tool calls per sample, with SVP calls decomposed into executor- and critic-side use. (c) ACC versus average token consumption.

63.86% to 67.73%. Although direct ReAct-style interaction is already effective for short factual questions, SVP retains the strongest overall performance, with F1 exceeding ReAct by 3.87 points. The fixed 300-example subset also shows clear performance separation across methods, providing a useful controlled comparison while keeping agentic evaluation tractable.

Overall, the strongest competing approach varies across benchmarks, while SVP consistently achieves the best performance. These results support the value of state-mediated system-level coordination for external knowledge integration across different reasoning settings.

4.3 Tool-Budget Scaling

The effectiveness of tool-augmented reasoning depends on how well an agent utilizes its available tools. We therefore analyze performance, tool utilization, and token consumption as the per-question budget increases on HLE-Verified Gold. Figure 2 compares ReAct and SVP under tool budgets ranging from 3 to 20. As shown in Figure 2(a), increasing the budget does not consistently improve ReAct, whose ACC changes from 48.05% to 47.46%. In contrast, SVP scales monotonically from 50.00% to 53.59%, expanding its advantage over ReAct from 1.95 to 6.13 points. This contrast shows that a larger tool budget alone is insufficient, while SVP more effectively translates additional tool capacity into performance gains.

Figure 2(b) further explains this difference. ReAct averages 0.46–1.42 calls per sample, using 7–15% of its budget. As the budget specifies only an upper bound rather than a search target, increasing it does not necessarily change ReAct's demand-driven tool use. The model may instead rely on its

Table 2: Performance of SVP and representative baselines across backbone LLMs on HLE-Verified Gold (668), measured by ACC (%).

Method	Qwen3.5-9B	Gemma4-12B-IT
DirectQA	21.11	32.04
ReAct	21.86	33.08
Plan + ReAct	<u>26.65</u>	<u>34.73</u>
Search-o1	22.60	33.38
Agentic Reasoning	26.05	32.49
SVP (Ours)	**33.53**	**42.07**

Table 3: Component analysis of SVP on HLE-Verified Gold (668), evaluating the effects of key framework design choices.

Variant	ACC	EM	F1
SVP w/ Critic	51.20	48.65	52.68
SVP w/o Critic	51.95	48.80	52.26
w/o State Box	49.85	47.60	51.36
Critic-Raw-Step	50.45	48.05	51.72
w/o Extractor	50.60	47.90	51.61
w/o Tools	49.70	47.60	51.07

parametric knowledge and stop searching once it considers the evidence sufficient. SVP averages 2.42–9.05 calls and maintains higher utilization of 45–81%. Its executor consistently invokes tools more frequently than ReAct, while critic-side calls provide additional evidence for progress assessment and control. Together with the scaling trend, these results show that SVP engages external knowledge more actively across step-local execution and system-level control.

As shown in Figure 2(c), SVP consumes more tokens than ReAct under the same budget because of its multi-stage orchestration. However, the additional computation is consistently accompanied by higher accuracy. Notably, SVP with a budget of 3 achieves 50.00% ACC using 72.7K tokens, outperforming ReAct with a budget of 20, which obtains 47.46% ACC using 91.8K tokens. Thus, SVP's advantage cannot be attributed solely to greater token consumption and is consistent with more effective orchestration of reasoning and external knowledge. The full numerical results across all tool budgets are provided in Appendix E.2.

4.4 Cross-Model Evaluation

To examine SVP's effectiveness across backbone LLMs, we evaluate it with two additional models, Qwen3.5-9B and Gemma4-12B-IT. As shown in Table 2, SVP performs best on both models, achieving 33.53% and 42.07% ACC, respectively. These results exceed the strongest baseline, Plan + ReAct, by 6.88 and 7.34 points. The consistent gains show that SVP's effectiveness extends beyond the primary DeepSeek-V4-Flash backbone across two additional model families.

Plan + ReAct ranks second with both backbones, suggesting that explicit task decomposition can benefit these models. However, its performance remains substantially below SVP, showing that fixed planning alone does not account for SVP's gains. ReAct and Search-o1 provide only modest improvements over DirectQA, while Agentic Reasoning varies across the two models. Overall, SVP maintains a clear advantage across all three evaluated backbone families. Detailed EM, F1, search-call, and token statistics are provided in Appendix E.3.

4.5 Component Analysis

We evaluate four design ablations and a no-tools control within SVP. *SVP w/o Critic* forces the final post-step action to Continue, disabling critic-triggered Refine Search and Replan while retaining the other SVP components. For the remaining variants, *w/o State Box* gives the executor only the current subgoal and removes the State Box from critic input, testing explicit cross-step state. *Critic-Raw-Step* gives the critic the full current-step reasoning context instead of the structured step view, testing structured critic input. *w/o Extractor* directly injects retrieved content, testing evidence extraction. *w/o Tools* disables web search to isolate the contribution of external knowledge.

As shown in Table 3, removing the State Box reduces ACC by 1.35 points, while replacing the structured step view with the raw step context and removing the extractor reduce ACC by 0.75 and 0.60 points, respectively. Disabling tools yields the largest ACC drop of 1.50 points, yet the resulting 49.70% ACC still exceeds the strongest baseline at 49.10%, indicating that SVP's orchestration remains effective even without web search.

The critic ablation reveals an interesting observation. SVP with and without critic-guided control achieve comparable aggregate performance, with differences of at most 0.75 points across ACC,

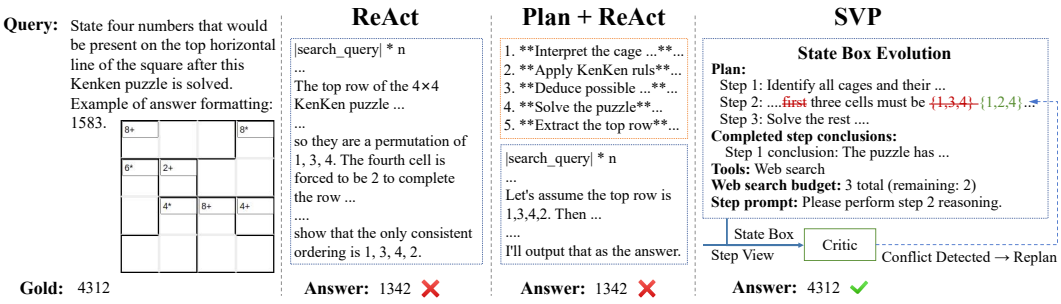

Figure 3: Case study of critic-triggered replanning on a 4×4 KenKen problem. ReAct and Plan + ReAct adopt an unsupported top-row ordering and return 1342. SVP detects a conflict between the current-step findings and the remaining plan, triggers replanning, and redirects subsequent reasoning to the correct answer 4312.

EM, and F1. These outcome-level metrics provide only a partial view of the critic's trajectory-level role. Recent work on agent evaluation similarly highlights that focusing only on final outcomes can overlook the step-by-step behavior of agentic systems, motivating evaluation of the intermediate task-solving process Zhuge et al. (2025). In SVP, the critic operates at this process level by assessing intermediate state, requesting additional evidence when needed, and triggering replanning when the current plan becomes inconsistent with accumulated evidence. Section 4.6 illustrates this behavior via a concrete trajectory, while Appendix E.4 reports intervention statistics showing that critic-guided control is actively exercised across a substantial fraction of trajectories during execution, and Appendix E.5 provides additional critic-guided plan-correction examples showing how the framework identifies unsupported or stale plan content, recommends replanning when needed, and redirects subsequent reasoning while preserving completed progress.

4.6 Case Study

Figure 3 compares the reasoning processes of ReAct, Plan + ReAct, and SVP on a 4×4 KenKen problem. ReAct repeatedly searches for relevant information but prematurely treats one permutation of the top-row candidates as the only valid ordering, producing 1342. Plan + ReAct first decomposes the task, yet its subsequent execution adopts the same unsupported ordering without revising the plan and reaches the same incorrect answer.

SVP also begins with an imperfect plan. After the first step, the state maintainer records the inferred cage structure as a compact conclusion for subsequent state construction and organizes the current claims, unresolved questions, and recent reasoning into a structured step view. By assessing this view together with the current State Box and accumulated evidence, the critic detects that the newly derived cage claims conflict with assumptions encoded in the remaining plan and requests additional evidence for verification. Because the additional evidence does not resolve the conflict, the critic recommends REPLAN, returning control to the planner. The planner preserves the completed conclusion while revising the remaining steps to correct the cage assignment and remove the premature ordering assumption. Guided by the revised State Box, the executor derives the correct answer 4312. Overall, this case illustrates how SVP uses accumulated state and critic-guided replanning to recover from an initially flawed trajectory by revising the remaining reasoning path.

5 Conclusion

We introduced SVP, a state-mediated framework for system-level external knowledge integration that uses explicit cross-step state to coordinate step-local ReAct execution with subsequent task orchestration. Experiments on three knowledge-intensive benchmarks show consistent gains over strong reasoning, tool-augmented, and agentic baselines, with analyses of tool-budget scaling, backbone models, key design choices, and reasoning behavior. These results support the value of state-mediated coordination for extending external knowledge integration beyond local reasoning. Promising directions include improving efficiency and control precision and extending SVP to dynamic environments where evidence, tools, or task conditions evolve during execution.

AI USE STATEMENT

Generative AI tools were used to assist with writing and language polishing, including improving the clarity, concision, and organization of author-written text. All technical content, experimental results, analyses, citations, and final wording were reviewed and verified by the authors. Generative AI was not used to generate or alter experimental results. The authors take full responsibility for the final content of this work, including any text produced with AI assistance.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the paper and supplementary materials. The Methodology section specifies the SVP framework and its components, while the Experimental Setup reports the evaluated models, decoding settings, search tools, tool budgets, datasets, and evaluation metrics. The Appendix provides additional implementation details, prompts, preprocessing and evaluation procedures, baseline configurations, component definitions, and extended experimental analyses. Anonymous supplementary materials include the implementation code and related configurations used in the experiments.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 2026.

Jingwen Deng, Jihao Huang, Zhen Hao Wong, Hao Liang, Quanqing Xu, Bin Cui, and Wentao Zhang. Data-centric perspectives on agentic retrieval-augmented generation: A survey. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1570–1588, 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge. *arXiv preprint arXiv:2509.07968*, 2025.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Moxin Li, Yong Zhao, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See Kiong Ng, Tat-Seng Chua, and Yang Deng. Knowledge boundary of large language models: A survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5131–5157, 2025a.

Weitao Li, Boran Xiang, Xiaolong Wang, Jingyi Ren, Ante Wang, Zhinan Gou, Weizhi Ma, and Yang Liu. Ur2: Unify rag and reasoning through reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12712–12751, 2026.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025b.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 120091–120131. Curran Associates, Inc., 2025c. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ae03bdef276132fae089692445725635-Paper-Conference.pdf.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Subhey Sadi Rahman, Md Adnanul Islam, Md Mahbub Alam, Musarrat Zeba, Md Abdur Rahman, Sadia Sultana Chowa, Mohaimenul Azam Khan Raiaan, and Sami Azam. Hallucination to truth: a review of fact-checking and factuality evaluation in large language models. *Artificial Intelligence Review*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28489–28503, 2025a.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. Hiprag: hierarchical process rewards for efficient agentic retrieval augmented generation. *arXiv preprint arXiv:2510.07794*, 2025b.

Zhao Xinjie, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. Reagent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop qa. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4067–4089, 2025.

Tianyang Xu, Dan Zhang, Kushan Mitra, and Estevam Hruschka. Verification-aware planning for multi-agent systems. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7528–7546, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Qijie You, Wenkai Yu, Hao Liang, Zhen Hao Wong, and Wentao Zhang. Agenticragtracer: A hop-aware benchmark for diagnosing multi-step retrieval reasoning in agentic rag. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1301–1335, 2026.

Weiqi Zhai, Zhihai Wang, Jinghang Wang, Boyu Yang, Xiaogang Li, Xander Xu, Bohan Wang, Peng Wang, Xingzhe Wu, Anfeng Li, et al. Hle-verified: A systematic verification and structured revision of humanity's last exam. *arXiv preprint arXiv:2602.13964*, 2026.

Xing Zhang, Yanwei Cui, Guanghui Wang, Wei Qiu, Ziyuan Li, Fangwei Han, Yajing Huang, Hengzhi Qiu, Bing Zhu, and Peiyang He. Verified multi-agent orchestration: A plan-execute-verify-replan framework for complex query resolution. *arXiv preprint arXiv:2603.11445*, 2026.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611, 2025.

A End-to-End SVP Procedure

Algorithm A.1 summarizes the end-to-end execution of SVP. Given a query and optional context, the planner generates two to five ordered high-level subgoals, while a global step limit L bounds execution and replanning. At each iteration, the executor solves only the current subgoal from the original task input and the current State Box. Executor- and critic-side tool calls share a per-question budget, and all retrieved content is processed by the extractor before use. The state maintainer then produces a compact conclusion for subsequent state construction and a structured step view for critic assessment. After optional evidence refinement, the critic issues CONTINUE or REPLAN, and the resulting action determines the State Box used for subsequent execution.

Algorithm A.1: End-to-End State-Verified Planning

Input: Query q, context x, tools $\mathcal{T}$, search budget B, step limit L
Output: Final answer y

$P^{(0)} \leftarrow \Pi(q, x)$
$C \leftarrow \emptyset, \quad E \leftarrow \emptyset$
$b \leftarrow B, \quad t \leftarrow 1, \quad k \leftarrow 0$
$H \leftarrow \mathrm{InitThread}(q, x, P^{(0)})$
while $t \leq \min\left(\left|P^{(k)}\right|, L\right)$ **do**
  $\rho_t \leftarrow P^{(k)}[t]$
  $S_t \leftarrow \left(P^{(k)}, C, \mathcal{T}, b, \rho_t\right)$
  $(\tau_t, E_t, m_t) \leftarrow \mathcal{E}(q, x, S_t; \mathcal{T}, b)$
  $E \leftarrow E \oplus E_t$
  $b \leftarrow b - m_t$
  $\bar{E}_t \leftarrow \mathcal{W}(E)$
  $(c_t, Z_t) \leftarrow \mathcal{M}(S_t, \tau_t, \bar{E}_t)$
  $C \leftarrow C \oplus (c_t)$
  $H \leftarrow H \oplus (S_t, \tau_t)$
  $(a_t, b, E) \leftarrow \mathrm{Assess}(S_t, Z_t, E, \mathcal{T}, b)$
  if $a_t = \textsc{Replan} \wedge t < L$ **then**
    $R_t \leftarrow \Pi_{\mathrm{rem}}(q, x, H)$
    $R_t \leftarrow \mathrm{ConstrainRemaining}(R_t, L - t)$
    $P^{(k+1)} \leftarrow P^{(k)}_{\leq t} \oplus R_t$
    $k \leftarrow k + 1$
    $H \leftarrow H \oplus \mathrm{PlanRecord}(P^{(k)})$
  end if
  $S_{t+1} \leftarrow \mathcal{U}(S_t, c_t, a_t; P^{(k)}, \mathcal{T}, b)$
  $t \leftarrow t + 1$
end while
$\Gamma_T \leftarrow \mathrm{AssembleAndTruncate}(H, C)$
$y \leftarrow \mathcal{A}(q, x, \Gamma_T)$
return y

Here, C stores compact conclusions from completed steps, while E stores accumulated extracted evidence separately from the State Box for critic assessment. The executor returns the step-local trajectory τ_t, extracted evidence E_t, and the number of tool calls m_t. The state maintainer produces the conclusion c_t and structured step view Z_t from the current trajectory and the bounded evidence window $\bar{E}_t$. The critic then assesses the current step from S_t, Z_t, and $\bar{E}_t$. The conclusion c_t enters the next State Box after the control action is applied.

The function Assess encapsulates critic-side validation and optional evidence refinement. It maintains E for evidence updates, while each critic assessment uses a bounded window $\mathcal{W}(E)$. When further evidence is needed, the critic issues an internal REFINE SEARCH request. The system executes the requested search, processes the results through the extractor, updates E and b, and the critic reassesses the same step with the updated evidence window. This is not a third cross-step action. Both CONTINUE and REPLAN retain c_t and advance execution. The former preserves the current plan, whereas the latter retains the completed prefix $P^{(k)}_{\leq t}$ and replaces only the remaining subgoals without re-executing the current step.

Replanning is constrained by the remaining slots under L. If the revised suffix exceeds this capacity, $\mathrm{ConstrainRemaining}$ uses the final available slot as a consolidation step for decisive evidence and unresolved assumptions. After termination, SVP assembles the recorded State Boxes and step-local trajectories for answer synthesis. When the context exceeds the model window, complete early step blocks are removed, while the query and planning preamble remain and the first retained State Box preserves conclusions from omitted steps. The no-tool variant follows the same procedure with executor- and critic-side search disabled.

B PROMPT TEMPLATES

This section presents the runtime prompt templates used by SVP. Text enclosed in braces denotes content inserted dynamically at inference time. Unless otherwise stated, the prompt text is reproduced verbatim from the implementation. For compact presentation, the role headers in the listings identify system and user messages but are not part of their content. Listings show the no-context form where task context is optional; the surrounding text specifies how context is inserted when present.

B.1 PLANNING AND REPLANNING PROMPTS

The planner receives the original query and optional task context, and returns two to five ordered high-level subgoals. The prompt constrains the planner to decompose the task without introducing unsupported factual assumptions.

Listing B.1: Initial planning prompt.

```
[SYSTEM]
You are a reasoning assistant. Your task is to analyze the question and
  create a high-level plan for how to solve it.

**Constraints:**
- Base your plan ONLY on the question structure and task decomposition.
- Do NOT assume any specific facts to be true (e.g. do not write 'Step
  1: X is 3' if X is not given).
- Think about what steps are needed and what kind of information might
  be required (including whether web search may be needed).

Output a single JSON object with a key "steps" whose value is a list of
  step strings, e.g.:
{"steps": ["Step 1: ...", "Step 2: ...", ...]}

Provide 2-5 main steps. Output nothing else except this JSON.

[USER]
Question:
{question}

Output the plan as JSON ({ "steps": [...] }):
```

When task context is available, a `Context:` block is inserted after the `Question:` header and before the question text.

Replanning is invoked only after the critic triggers a REPLAN action. The replanner receives the original task input and the accumulated main-thread history, and generates only the remaining subgoals. The completed prefix is preserved by the surrounding control procedure rather than regenerated by the replanner.

Listing B.2: Remaining-plan revision prompt.

```
[SYSTEM]
You are a reasoning assistant. Given the full conversation below (
  planning and reasoning so far), output a new plan for the **remaining
  ** steps only. Base steps on the question and what has been done; do
  not assume unverified facts.

Output a single JSON object: {{ "steps": ["Step 1: ...", "Step 2: ...",
  ...] }}. Output nothing else.

[USER]
Question:
{question}

**Full conversation so far:**
```

```
{main_thread_text}

Output the remaining steps as JSON ({{ "steps": [...] }}):
```

When present, a `Context:` block is inserted after the `Question:` header and before the question text. The double braces in Listing B.2 are reproduced literally from the runtime prompt. If the returned suffix exceeds the remaining execution slots, the control procedure truncates it and replaces the final available slot with the consolidation step described in Algorithm A.1.

B.2 Executor and Extractor Prompts

At step t, the executor receives the original query, optional context, and current State Box. It is instructed to solve only the current subgoal, invoke web search when needed, and avoid producing the final answer for the complete task.

Listing B.3: Step-local executor prompt in the tool-enabled setting.

```
[SYSTEM]
You are a reasoning assistant. Execute the current subgoal step by step.

**Available tools:** {tools_str}

To perform a web search, write: <|begin_search_query|>your search query
  <|end_search_query|>
Results will appear in <|begin_search_result|>...<|end_search_result|>.

At the **end** of your reasoning for this step, you **must** output a
  short conclusion inside: <step_summary>...</step_summary>
The content must be one or two sentences stating the key finding or
  conclusion of this step, not a plan or search intent.
State the step conclusion based on the reasoning/evidence in this step;
  avoid generic conceptual restatements.
Do not output the final answer to the whole question yet.

[USER]
**Question:**
{question}

**Context:**
{context}

{state_box}

Execute the subgoal above. Use web search when you need external or
  factual information:
```

For the no-tool variant, the available-tool description and search syntax are removed, while the step-local execution and step-summary requirements remain unchanged.

After an executor-side search, the retrieved content is processed by the extractor and returned to the same step through the following continuation message:

Listing B.4: Search-result continuation message for step-local execution.

```
[USER]
<|begin_search_result|>
[Web search budget] Remaining calls: {remaining}/{total}

{extracted_facts}
<|end_search_result|>

Continue your reasoning for this step. Do not output the final answer
  yet.
```

The extractor receives the search query and formatted Serper–Jina results. It returns a JSON object containing a list of concise facts with source URLs and confidence labels.

Listing B.5: Evidence-extraction prompt.

```
[SYSTEM]
You are a fact extractor. Given a search query and raw web search
  results, extract only the key facts that directly help answer the
  query. Be extremely concise.

Extraction objective (important):
- Prefer concrete, fillable details from the provided content (numbers,
  dates, names, units, definitions, conditions, formulas, thresholds,
  rankings).
- If the query expects a specific value (e.g., 'what is the value?', '
  how much?', 'when?', 'rank?', 'threshold?'), output that value
  explicitly (and units when present).
- Prefer copying a short exact phrase (quote-first) from the 'Content:'
  portion that contains the value; do NOT invent values.
- Avoid high-level summaries like 'This article discusses ...' unless
  the query itself is about a general concept.
- If the requested specific value is not present in the provided
  snippets, output a low-confidence fact like 'Not found in provided
  search snippets.'
- Do NOT include any <|begin_...|> / <|end_...|> tags inside the fact
  text.

Output a JSON object: {"facts": [{"fact": "...", "source_url": "...", "
  confidence": "high|medium|low"}, ...]}

Constraints:
- Each fact: one short sentence, ≤250 tokens
- Total output: ≤1200 tokens
- Include source_url from the search results
- confidence: high (exact or very explicit value present in content),
  medium (plausible paraphrase), low (missing / vague / indirect)

Output nothing else except valid JSON.

[USER]
Query: {search_query}

Raw search results:
{raw_formatted_results}

Extract key facts as JSON. Discard irrelevant content. Output only valid
   JSON:
```

When the search backend returns no usable organic result, the implementation bypasses the extractor and inserts a low-confidence search-failure record instead.

B.3 State Maintenance and Critic Prompts

State maintenance uses two separate LLM calls. First, an independent step summarizer produces the compact conclusion c_t stored in subsequent State Boxes. This is distinct from the <step_summary> text produced within the executor response.

Listing B.6: Compact step-conclusion prompt.

```
[SYSTEM]
You are a summarizer. Given the **current step** reasoning text below (
  which may include tool/search results and multi-turn reasoning),
  output a **short, informative summary** (1-3 sentences) that captures
  the key findings, facts, or conclusions from this step. Be concise and
```

```
   useful for subsequent steps; do not repeat the subgoal verbatim.
  State concrete conclusions grounded in this step's reasoning/evidence;
   avoid generic conceptual restatements. When available, prioritize
  answer-bearing details (e.g., specific values, dates, entities,
  thresholds, or direct comparisons) and include brief qualifiers when
  uncertainty remains. Output only the summary text, no labels or JSON.

[USER]
**Current subgoal for this step:**
{subgoal}

**Reasoning text for this step:**
{reasoning_text}

Output a short, informative summary (1-3 sentences) of the key
  conclusions/facts from this step:
```

Second, the state extractor derives the claims and unresolved questions used in the structured step view Z_t. The evidence summary is constructed from a bounded recent window of accumulated evidence.

Listing B.7: Structured state-extraction prompt.

```
[SYSTEM]
You are an extractor. From the given reasoning text and evidence, output
   a single JSON object with two keys:
- derived_claims: a list of strings, each a concise claim or conclusion
  that is supported by the reasoning/evidence
- open_questions: a list of strings, each a sub-question that is still
  not answered
Output nothing else except this JSON. Example: {"derived_claims":
  ["...", "..."], "open_questions": ["..."]}

[USER]
**Subgoal:**
{subgoal}

**Evidence (summary):**
{evidence_summary}

**Reasoning text:**
{reasoning_text}

Output JSON with derived_claims and open_questions:
```

The critic receives the current subgoal and a rendered state view containing the current State Box, a bounded recent evidence window, current-step derived claims and open questions, and a bounded recent step trace. It returns a structured recommendation used by the control loop.

Listing B.8: Critic-assessment prompt.

```
[SYSTEM]
You are a verifier. Given a subgoal and the current state (evidence and
  claims), determine whether the subgoal is satisfied. Do not generate
  new knowledge.

Output a single JSON object with exactly these keys:
- is_subgoal_satisfied: true or false
- missing_evidence: list of strings describing what evidence is still
  missing (empty if satisfied)
- unsupported_claims: list of claim strings that lack evidence support (
  empty if none)
- recommendation: one of "continue", "replan", "refine_search"
```

```
Use "continue" when the subgoal is satisfied. Use "refine_search" when
  key evidence is missing. Use "replan" when the approach needs to
  change.

Output nothing else except this JSON.

[USER]
**Subgoal:**
{subgoal}

**Current state:**
{state_summary}

Output JSON:
```

A `refine_search` recommendation is handled within the current-step critic loop. The system executes the requested search, processes the retrieved content through the same extractor, appends the extracted evidence to the accumulated evidence, and the critic reassesses the current step. Only CONTINUE and REPLAN act as cross-step control decisions.

B.4 ANSWER SYNTHESIS PROMPT

The answer-synthesis input is assembled directly in the SVP runner rather than through a separate prompt-manager function. A dedicated system message is followed by the truncated main thread, with historical system messages removed, and a final user instruction requesting the boxed answer.

Listing B.9: Answer-synthesis message assembly in the tool-enabled setting.

```
[SYSTEM]
You are a reasoning assistant with access to web search. The
  conversation below contains the original question, plan, and step-by-
  step reasoning. Your role is to follow this reasoning process. When
  asked for the final answer, output only in the format \boxed{
  YOUR_ANSWER}. Output nothing else after the boxed answer.

[TRUNCATED MAIN THREAD]
{original question and context}
{planning history}
{retained State Boxes and step-local trajectories}

[USER]
Based on the complete reasoning above, provide the final answer in \
  boxed{...}:
```

In the no-tool variant, the phrase "with access to web search" is removed from the system message. The main thread is truncated by complete step blocks when necessary, while the task preamble is retained and the first remaining State Box preserves compact conclusions from omitted steps. These retained blocks are assembled only for final answer synthesis rather than propagated between executor steps.

C DETAILED EXPERIMENTAL SETUP

C.1 MODEL AND INFERENCE CONFIGURATION

DeepSeek-V4-Flash (0424), accessed through SiliconFlow, serves as the primary reasoning backbone. Within SVP, the same model is used for planning, step-local execution, evidence extraction, state maintenance, critic assessment, replanning, and answer synthesis. The compared methods also use this backbone in the primary experiments.

We set the temperature to 0.1 and the maximum output length to 16,384 tokens per LLM call. The latter is a per-call limit rather than a total trajectory budget. Thinking mode remains enabled

for reasoning-based methods. The main-table *DirectQA no thinking* baseline explicitly disables thinking, while DirectQA in the cross-model evaluation follows each local model's default thinking configuration. Each request uses a timeout of 1,800 seconds. SVP permits at most ten executed steps per question and up to three critic-side refinement rounds within each step. Each reported result is based on a single complete evaluation run over the corresponding dataset under the specified configuration.

For image-based HLE questions, Kimi K2.6 converts each image into a textual description before reasoning begins, using a temperature of 0.1 and a maximum output length of 768 tokens. The descriptions are cached and reused across methods, ensuring identical textual inputs. The vision model does not participate in subsequent reasoning or answer generation.

For cross-model evaluation, Qwen3.5-9B and Gemma4-12B-IT are served with vLLM on NVIDIA H100 GPUs. We use their corresponding chat templates and recommended decoding settings while keeping the evaluation data, prompts, search backend, and search budget unchanged.

C.2 Search and Tool Configuration

All tool-augmented methods use Serper for web search and Jina Reader for webpage extraction. Each search returns at most ten results, and retrieved webpage content is truncated to 6,000 characters. When webpage retrieval fails, the corresponding search snippet is used as a fallback.

The default search budget is five calls per question. Each Serper request consumes one budget unit, whereas webpage reading does not consume an additional call. In SVP, executor-side searches and critic-triggered refinement searches share this global budget. The tool-budget experiment changes only this limit:

$$B \in \{3, 5, 7, 10, 20\}.$$

The search backend and budget are standardized, while each method retains its own evidence-processing procedure. ReAct and Plan + ReAct receive formatted search results directly, whereas SVP processes retrieved content through its extractor. Search-o1, CRITIC, and Agentic Reasoning retain their respective retrieval and feedback mechanisms.

C.3 Baseline Implementation

All methods share the same evaluation pipeline, primary backbone, dataset inputs, answer extraction, and evaluation procedure. Tool-augmented methods additionally share the same search backend and search budget.

No-External-Knowledge Baselines. *DirectQA no thinking* directly produces the final answer without explicit reasoning, planning, or tool access, with thinking disabled. *CoT* uses the backbone's thinking mode to perform step-by-step reasoning without external tools. *Plan + CoT* first generates a high-level plan and then reasons without tools under this fixed plan. In the cross-model evaluation, DirectQA instead retains each local model's default thinking configuration.

Module-Level Baselines. *ReAct* interleaves reasoning with web search in a single trajectory. *Plan + ReAct* adds an initial high-level plan that remains fixed during execution. *CRITIC* performs tool-assisted critique and answer revision, while *Search-o1* integrates iterative search and document-level evidence extraction into reasoning. Their original control mechanisms are retained under the shared model, search backend, and budget.

System-Level Baseline. *Agentic Reasoning* retains its multi-stage reasoning, web-search, and structured-memory workflow. For a controlled web-search comparison, its additional code-execution tool is disabled. Its temperature is set to 0.1, consistent with the other main-table methods.

SVP Variants. Full SVP includes planning, step-local execution, evidence extraction, state maintenance, critic assessment, and replanning. *w/o State Box* gives the executor only the current subgoal and removes the State Box from critic input. *w/o Critic* forces the final post-step action to CONTINUE, disabling critic-triggered REFINE SEARCH and REPLAN while retaining the other SVP

components. *Critic-Raw-Step* replaces the structured step view with the full current-step reasoning context for critic assessment. *w/o Extractor* directly injects retrieved content without evidence extraction. *w/o Tools* preserves the orchestration procedure but disables web search.

D Datasets and Evaluation

D.1 Dataset Details and Preprocessing

HLE-Verified Gold. We evaluate all 668 examples in the Gold subset of HLE-Verified, covering diverse expert-level domains and both textual and image-based questions. Images are converted into cached textual descriptions using the preprocessing procedure described above.

GPQA-Diamond. We use the complete 198-example test set, consisting of graduate-level multiple-choice questions in biology, chemistry, and physics.

SimpleQA-Verified-300. SimpleQA Verified contains 1,000 short-form factual questions. To keep repeated multi-method agentic evaluation tractable, we use the first 300 examples in the released file order as a fixed, reproducible subset. This subset is a deterministic prefix rather than a random or identifier-based selection. Reference URLs included with the dataset are withheld from all methods and are neither inserted into prompts nor fetched automatically.

D.2 Evaluation Procedure

Models are instructed to return the final answer inside `\boxed{...}`. The evaluator extracts the last complete boxed expression using balanced-brace parsing. When no boxed answer is found, task-specific fallback patterns are used to identify an explicit final answer or option letter.

For open-ended questions, predictions and references are normalized by lowercasing, stripping leading and trailing whitespace, and collapsing consecutive whitespace into a single space. No additional article or punctuation removal is applied. Let p and g denote the normalized prediction and reference, respectively. We compute Exact Match (EM) and Accuracy (ACC) for each example as

$$\mathrm{EM}(p, g) = \mathbf{1}[p = g], \qquad \mathrm{ACC}(p, g) = \mathbf{1}[g \sqsubseteq p],$$

where $g \sqsubseteq p$ indicates that the normalized reference string g occurs as a substring of the normalized prediction p. For token-level F1, let P and G denote the whitespace-token multisets of p and g, and let $c_P(t)$ and $c_G(t)$ be the multiplicities of token t. Their overlap is

$$n_{\text{same}} = \sum_t \min(c_P(t), c_G(t)) \,.$$

Precision and recall are then

$$\mathrm{Prec} = \frac{n_{\text{same}}}{|P|}, \qquad \mathrm{Rec} = \frac{n_{\text{same}}}{|G|},$$

and

$$\mathrm{F1} = \begin{cases} 0, & n_{\text{same}} = 0, \\ \dfrac{2\,\mathrm{Prec}\,\mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}}, & \text{otherwise.} \end{cases}$$

For multiple-choice questions, explicit option letters are extracted directly; when option text is returned, it is conservatively mapped to the corresponding choice. Dataset-level ACC, EM, and F1 are obtained by averaging the corresponding per-example scores and are reported as percentages.

All examples remain in the evaluation denominator, and failed or unextractable responses are scored as incorrect. Average search calls count actual Serper requests per question. Average token consumption sums input and output tokens across the main reasoning-model calls, excluding separately recorded vision-model tokens.

Table 4: Full benchmark results using DeepSeek-V4-Flash. ACC, EM, and F1 are percentages. Search denotes average Serper calls per question, and Tok. denotes average main-LLM tokens in thousands. NT denotes disabled thinking, and Agentic denotes Agentic Reasoning.

Method	ACC	EM	F1	Search	Tok. (K)
HLE-Verified Gold (668)					
DirectQA (NT)	29.94	28.59	31.44	0.00	0.9
CoT	47.75	45.66	48.73	0.00	14.3
Plan + CoT	46.86	44.16	47.69	0.00	15.4
ReAct	48.65	45.36	49.00	0.61	27.3
Plan + ReAct	46.71	43.71	46.97	0.37	24.0
CRITIC	46.86	42.22	45.34	< 0.01	134.3
Search-o1	49.10	45.81	49.17	0.77	12.0
Agentic	45.51	42.37	45.32	0.95	22.7
SVP	**51.20**	**48.65**	**52.68**	3.68	83.7
GPQA-Diamond (198)					
DirectQA (NT)	69.19	69.19	69.19	0.00	0.7
CoT	84.85	84.85	84.85	0.00	10.1
Plan + CoT	84.34	84.34	84.34	0.00	11.1
ReAct	83.84	83.84	83.84	0.63	28.5
Plan + ReAct	81.31	81.31	81.31	0.36	20.0
CRITIC	80.81	80.30	80.30	0.00	99.2
Search-o1	78.79	78.79	78.79	0.27	7.4
Agentic	80.81	80.81	80.81	0.23	10.2
SVP	**85.35**	**85.35**	**85.35**	3.39	82.6
SimpleQA-Verified-300					
DirectQA (NT)	16.67	15.67	25.47	0.00	0.1
CoT	21.33	19.67	30.21	0.00	2.3
Plan + CoT	20.33	18.67	29.67	0.00	2.9
ReAct	54.33	50.00	63.86	1.83	50.2
Plan + ReAct	53.00	46.00	62.94	1.85	53.0
CRITIC	22.67	19.00	31.51	0.45	50.8
Search-o1	36.00	31.33	45.69	3.15	11.4
Agentic	15.00	13.00	21.31	4.46	72.1
SVP	**55.00**	**50.67**	**67.73**	4.42	54.7

E Extended Experimental Results

E.1 Full Benchmark Results

Table 4 supplements the main results with average search calls and token consumption. The additional statistics show that resource consumption alone does not explain the performance differences across methods. For example, CRITIC consumes substantially more tokens than several stronger methods on HLE-Verified Gold and GPQA-Diamond. SVP incurs additional computation from its multi-stage orchestration, while achieving the strongest answer-quality results across all three benchmarks.

E.2 Full Tool-Budget Results

Table 5 provides the exact values underlying the tool-budget analysis in the main paper. ReAct continues to use only a small fraction of the available search budget, while its token consumption increases without a consistent improvement in answer quality. SVP performs progressively more searches as the budget increases and improves from 50.00% to 53.59% ACC.

The additional calls are used primarily by the executor. From $B = 3$ to $B = 20$, average executor-side search increases by 5.67 calls, compared with 0.96 additional critic-triggered refinement searches. This indicates that larger budgets mainly expand step-local evidence acquisition rather than critic-side verification.

Table 5: Full tool-budget results on HLE-Verified Gold (668). B is the per-question search budget. ACC, EM, and F1 are percentages; Search and Tok. denote average search calls and main-LLM tokens, respectively.

Method	B	ACC	EM	F1	Search	Tok. (K)
ReAct	3	48.05	45.06	48.51	0.46	20.5
	5	48.65	45.36	49.00	0.61	27.3
	7	46.86	44.46	47.68	0.90	43.9
	10	47.60	44.31	47.38	1.02	54.4
	20	47.46	45.21	48.65	1.42	91.8
SVP	3	50.00	47.60	50.91	2.42	72.7
	5	51.20	48.65	52.68	3.68	83.7
	7	51.35	48.50	51.87	4.85	90.0
	10	51.95	49.70	53.09	5.86	98.6
	20	**53.59**	**50.90**	**54.41**	9.05	151.1

Table 6: Detailed cross-model results on HLE-Verified Gold (668). DirectQA follows each local backbone's default thinking configuration. ACC, EM, and F1 are percentages; Search and Tok. denote average search calls and main-LLM tokens, respectively.

Method	ACC	EM	F1	Search	Tok. (K)
Qwen3.5-9B					
DirectQA	21.11	20.81	22.14	0.00	14.1
ReAct	21.86	21.11	23.44	0.36	13.8
Plan + ReAct	26.65	26.05	28.53	0.33	15.7
Search-o1	22.60	22.01	23.27	1.01	13.9
Agentic	26.05	25.30	27.34	0.98	24.2
SVP	**33.53**	**32.04**	**35.40**	4.46	138.3
Gemma4-12B-IT					
DirectQA	32.04	31.29	33.85	0.00	11.0
ReAct	33.08	32.34	35.25	0.06	10.7
Plan + ReAct	34.73	33.83	36.43	0.07	15.1
Search-o1	33.38	32.63	35.21	0.28	10.6
Agentic	32.49	31.74	34.53	0.41	16.3
SVP	**42.07**	**40.27**	**44.06**	1.27	113.3

Table 7: Critic intervention statistics on HLE-Verified Gold (668). Assessment-level rates are computed over 3,652 critic assessments; trajectory-level events are computed over 668 samples and may overlap.

Statistic	Count	Rate
Critic assessment outcomes		
CONTINUE	2,195	60.1%
REFINE SEARCH	1,109	30.4%
REPLAN	348	9.5%
Trajectory-level intervention coverage		
Any REFINE SEARCH request or REPLAN action	420	62.9%
Any REPLAN action	228	34.1%
REFINE SEARCH before later REPLAN	166	24.9%

E.3 DETAILED CROSS-MODEL RESULTS

Table 6 extends the cross-model evaluation in the main paper with EM, F1, average search calls, and token consumption, using the same set of compared methods for both local backbones.

E.4 CRITIC INTERVENTION STATISTICS

Table 7 summarizes how the critic participates in closed-loop control on HLE-Verified Gold under the default tool budget. Across 3,652 assessments, the critic returns CONTINUE in 60.1% of cases, requests REFINE SEARCH in 30.4%, and recommends REPLAN in 9.5%. At the trajectory level, 62.9% of samples contain at least one REFINE SEARCH request or REPLAN action, and 34.1% undergo at least one REPLAN. These statistics show that critic-guided control is actively exercised during execution rather than serving only as a passive verification stage.

E.5 ADDITIONAL CRITIC-GUIDED PLAN CORRECTIONS

The aggregate ablation in Section 4.5 provides only an outcome-level view of critic-guided control. To make its trajectory-level role more concrete, we present three additional HLE-Verified trajectories in which accumulated step conclusions no longer support assumptions encoded in the remaining plan. In each case, the critic flags unsupported content and recommends REPLAN; control then returns to the planner, which revises the remaining steps and redirects subsequent reasoning toward the correct answer. The first two examples show direct replanning from the existing state, whereas the third shows evidence refinement followed by replanning. For clarity, we show only the decisive parts of the State Box and omit unchanged or irrelevant content with "...". Because replanning preserves completed progress and revises only the remaining plan, outdated wording may remain in already completed plan steps; the excerpts below emphasize the revised suffix.

Example 1: Correcting an inconsistent mutation-spectrum plan. Sample ID: `673a8c2e4e2e35b51a27fafd`. The question asks whether hypermutator strains of *Pseudomonas aeruginosa* produce a wider or narrower spectrum of *mucA* mutations. The accumulated conclusions already support a narrower mutation spectrum, while the remaining plan still assumes a wider spectrum and would therefore select the wrong option.

```
[Before Replan]

Plan:
  ...
  Step 3: Consider the spectrum of mutations: with more
          mutations, a wider variety of mucA mutations
          ... is expected, not just a narrow set.
  Step 4: Compare with answer choices: select the option
          that states increased frequency and wider
          spectrum of mucA mutations.

Completed step conclusions:
  Step 1: ... mutations in mucA are predominantly
          restricted to simple sequence repeat deletions,
          resulting in a narrower spectrum ...
  Step 2: ... mutations in mucA are predominantly limited
          to specific types ..., resulting in a narrower
          spectrum ...
```

The critic flags the wider-spectrum claim as unsupported and recommends REPLAN. The planner therefore revises the remaining decision step to follow the narrower-spectrum conclusion already recorded in the State Box. Notably, the search budget is already exhausted at this point, so the correction is made by reconciling the remaining plan with the existing state rather than by acquiring additional evidence.

```
[After Replan]

Plan:
  ...
  Step 4: Compare with answer choices: select the option
          that states increased frequency and narrower
          spectrum of mucA mutations.

Completed step conclusions:
```

```
  ...
  Step 3: Increased mutagenesis ... raises the frequency
          of mucoid variants, but the mucA mutations are
          predominantly limited to simple sequence repeat
          deletions, yielding a narrower spectrum ...
```

The revised remaining plan is now consistent with the accumulated conclusions and leads to the correct answer, option C.

Example 2: Revising an outdated temperature estimate. Sample ID: `66febb7d2cb8fd8d04bcce70`. A similar inconsistency appears in an image-based question asking for the martensite start temperature of an LTTE weld filler. The completed-step conclusions place the transition between $150°C$ and $200°C$, whereas the remaining plan still targets $100°C$–$150°C$ and option B.

```
[Before Replan]

Plan:
  ...
  Step 3: ... the key transition occurs between 100 C
          and 150 C ..., so Ms is likely in the range
          100 C--150 C.
  Step 4: Match the inferred Ms range to the answer
          choices. The range 100 C--150 C corresponds
          to option B.

Completed step conclusions:
  Step 1: ... the martensite start temperature (Ms)
          lies between 150 C and 200 C ...
  Step 2: ... Ms lies between 150 C and 200 C,
          likely near the point where the stress
          pattern shifts.
```

The critic flags the $100°C$–$150°C$ estimate as unsupported by the current state and recommends REPLAN. The planner then revises the remaining answer-selection step rather than discarding the completed progress.

```
[After Replan]

Plan:
  ...
  Step 4: Match the inferred Ms range (150 C--200 C)
          to the answer choices. This corresponds to
          option F.

Completed step conclusions:
  ...
  Step 3: ... the martensite start temperature (Ms)
          lies between 150 C and 200 C.
```

The revised remaining plan selects option F, matching the reference answer.

Example 3: Evidence refinement followed by replanning. Sample ID: `671579d27c48af0286fef21b`. The third example concerns the fundamental group of a topological space formed by gluing two pairs of pants and identifying their waistbands. Here, the accumulated state already supports $(\mathbb{Z} \times \mathbb{Z}) * \mathbb{Z}$, while the original plan continues toward $\mathbb{Z} \times \mathbb{Z}$. Unlike the first two cases, the critic first requests additional evidence through REFINE SEARCH. After the requested searches fail to support the original route, subsequent critic assessment recommends REPLAN.

```
[Before Replan]

Plan:
```

```
  Step 1: ... the space is a torus with two holes ...
  Step 2: Compute the fundamental group of the torus
          with two holes: It is a free group on three
          generators ...
  Step 3: ... we deduce that the resulting fundamental
          group is ... Z x Z.
  Step 4: Match with answer choices: Z x Z is option I.

Completed step conclusions:
  Step 1: ... the space is homotopy equivalent to a
          torus with two points identified, whose
          fundamental group is the free product
          (Z x Z) * Z.
```

After evidence refinement, the critic continues to flag the route based on the wedge-of-three-circles/free-group interpretation as unsupported and recommends REPLAN. The planner then replaces the remaining derivation with one consistent with the accumulated state.

```
[After Replan]

Plan:
  ...
  Step 3: ... the space is homotopy equivalent to a
          torus with two points identified. Using
          Van Kampen, the fundamental group becomes
          the free product of the torus fundamental
          group (Z x Z) with Z, i.e., (Z x Z) * Z.
  Step 4: Match with answer choices:
          (Z x Z) * Z is option J.

Completed step conclusions:
  Step 1: ... fundamental group is (Z x Z) * Z.
  Step 2: ... the two boundary loops are represented
          by c and [a,b]c^{-1} ...
```

The replanned trajectory therefore replaces the previous option-I route with option J, which is the correct answer.

Across these examples, the critic acts as a verifier and controller over the evolving reasoning state rather than directly rewriting the plan itself. It identifies unsupported content and recommends replanning, after which the planner revises the remaining suffix while preserving completed progress. These traces complement the aggregate ablation by exposing a trajectory-level role that final-answer metrics alone do not show: critic-guided control can detect stale or unsupported plan content and redirect subsequent reasoning toward a correct solution.