EFFICIENT AGENTIC REASONING THROUGH SELF-REGULATED SIMULATIVE PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

How should an agent decide when and how to plan? A dominant approach builds the agent as a reactive policy with adaptive computation (e.g., chain-of-thought reasoning), trained end-to-end with the expectation that planning will emerge implicitly from sufficient data and compute. Without control over the presence, structure, or horizon of planning, however, these systems typically increase reasoning length dramatically during training, leading to inefficient token consumption that does not reliably translate to accuracy gains. We argue that efficient agentic reasoning benefits from a decomposition of decision-making into three interacting systems: **simulative reasoning** (System II) that grounds deliberation in future-state prediction using a world model, rather than unconstrained chain-of-thought; **self-regulation** (System III) that decides when and how deeply the agent plans at each turn through a learned **configurator**; and **reactive execution** (System I) that handles fine-grained reasoning and action. To test this, we develop SR^{2}AM (Self-Regulated Simulative Reasoning Agentic LLM), which realizes the configurator and simulative planning as distinct stages within an LLM's chain-of-thought reasoning, with the LLM itself serving as the world model in language space. We explore two implementations, v0.1-8B and v1.0-30B, both trained via supervised learning followed by reinforcement learning (RL). Across mathematical reasoning, scientific problem-solving, tabular data analysis, and web information seeking, SR^{2}AM-v0.1-8B and SR^{2}AM-v1.0-30B achieve overall Pass@1 competitive with systems at 120–357B and 685B–1T parameters, respectively, while SR^{2}AM-v1.0-30B consumes 25.8–94.9% fewer reasoning tokens than strong agentic LLMs of similar scale. Analysis reveals that RL increases average planning horizon by 22.8% while planning frequency grows only 2.0 percentage points, indicating that the model learns to plan *further ahead* rather than *more often*.

1 INTRODUCTION

A long-standing goal of AI is to build agents capable of long-horizon planning and goal-oriented behavior (McCarthy et al., 1955; Newell et al., 1959). Across recent embodied and language-based systems, a common approach has emerged: treat the agent as a reactive policy with possibly adaptive computation (e.g., chain-of-thought (Wei et al., 2022) for large language models (LLMs, Achiam et al., 2023), latent conditioning for vision-language-action models (VLAs, Figure AI, 2025; Physical Intelligence, 2024)), and train it end-to-end with the expectation that planning capabilities will emerge implicitly from sufficient data, compute, and task training. Current agentic LLMs are a prominent instantiation of this philosophy. These systems deploy reasoning models to think and act via unconstrained chain-of-thought (CoT; Wei et al., 2022; Yao et al., 2023b), sometimes refined with reinforcement learning (RL) for task success (DeepSeek-AI, 2025; Jin et al., 2025). By interacting with environments consisting of tools and predefined interaction logic (e.g., Agent Skills (Zhang et al., 2025a)), they can solve challenging problems in web browsing (OpenAI, 2025b; Steinberger, 2026), software engineering (Anthropic, 2025; OpenAI, 2025a), STEM reasoning (OpenAI, 2024a; DeepSeek-AI, 2025), and deep research (Google, 2024; OpenAI, 2025f), going beyond what parametric knowledge (e.g., Achiam et al., 2023) and single-pass reasoning (e.g., DeepSeek-AI, 2025) can afford.

Coherent long-horizon behavior, however, requires deliberate planning, and yet the dominant approach falls short in a fundamental way. Planning is expected to emerge within undifferentiated chain-of-

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.

thought, with no mechanism to control its presence, horizon, or structure. Without control over *what* the model reasons about, token consumption increases dramatically during training, while longer reasoning does not necessarily yield better answers (Gema et al., 2025; Su et al., 2025). More broadly, this approach provides no explicit planning structure that can be analyzed, regulated, or improved independently of the rest of the reasoning process.

We argue that efficient agentic reasoning benefits from decomposing deliberation into three interacting systems: **reactive execution** (System I) for fine-grained reasoning and direct action; **simulative reasoning** (System II) that predicts consequences of proposed actions through a world model, providing a unified planning mechanism across diverse tasks (Xing et al., 2025); and **self-regulation** (System III) that decides *when* and *how deeply* to plan through a learned **configurator**, much like humans modulate deliberation based on urgency, uncertainty, and complexity (Kahneman, 2011). Prior efforts address part of this problem: controlling reasoning amount (e.g., Lou et al., 2025; Wang et al., 2025a), selecting execution mode at task onset (Chen et al., 2025b; Jiang et al., 2025), distilling rule-based workflows (Li et al., 2025c), or using world models for obligatory simulation (e.g., Hao et al., 2023; Deng et al., 2025)). None combines all three into a unified architecture.

In this paper, we study whether the System I+II+III decomposition yields better accuracy-efficiency tradeoffs than unregulated or partially regulated alternatives, in the setting of language-based interactive reasoning (e.g., Mathematical Association of America, 2024; Wei et al., 2025). To test this, we develop SR^{2}AM (Self-Regulated Simulative Reasoning Agentic LLM), which implements the configurator and simulative planner as distinct stages within an LLM's CoT reasoning, with the LLM itself serving as the world model. At each turn, the configurator (System III) assesses the current state and decides how to proceed (e.g., make a new plan, continue an existing one, or act directly); when invoked, the simulative planner (System II) plans by proposing actions and predicting future states. These components operate alongside free-form reasoning and acting (System I), separating self-regulation, planning, and execution while preserving end-to-end expressiveness.

Specifically, we explore two implementations: v0.1, which records decisions from a multi-module prompted system to demonstrate feasibility, and v1.0, which reconstructs structured plans from pretrained reasoning LLM traces for better scalability. Both are trained via supervised learning followed by RL, yielding SR^{2}AM-v0.1-8B and SR^{2}AM-v1.0-30B, respectively. In evaluations on interactive reasoning for math, science, tabular analysis, and web information seeking, SR^{2}AM-v0.1-8B and SR^{2}AM-v1.0-30B achieve overall Pass@1 competitive with systems at 120–357B and 685B–1T parameters, respectively, while SR^{2}AM-v1.0-30B consumes 25.8–94.9% fewer reasoning tokens than strong agentic LLMs of similar scale. Analysis shows that RL increases average planning horizon by 22.8% while planning frequency grows only 2.0 percentage points, indicating the model learns to plan *further ahead* rather than *more often*.

2 Formalizing Self-Regulated Simulative Reasoning

2.1 Agent-Environment Model and Simulative Reasoning

Consider a sequential interaction between an agent and an environment. At time step t, the agent π outputs action a_t given world state s_t, and the universe μ transitions to the next state s_{t+1} according to $p_\mu(s_{t+1} \mid s_t, a_t)$. The agent receives reward $r(s_t, g)$ based on its goal g, and aims to maximize its value function $V^g_{\pi,\mu}(s_t) = \mathbb{E}_{\pi,\mu}\left[\sum_{k=t}^{\infty} \gamma_k r(s_k, g) \mid s_t\right]$ (Sutton & Barto, 1998) by *planning* action sequences $a^*_t, a^*_{t+1}, \ldots$ that account for both immediate reward and predicted future states $s_{t+1}, s_{t+2}, \ldots$ Beyond simple fully observable settings (e.g., Silver et al., 2016; 2018), however, the agent does not have direct access to the true world state s_t. Instead, it receives observations o_t and infers a *belief state* $\hat{s}_t$. A *world model* f can predict the next belief state $\hat{s}_{t+1}$ given a proposed action a'_t, according to $p_f(\hat{s}_{t+1} \mid \hat{s}_t, a'_t)$. By simulating sequences of actions and their predicted consequences, the agent can approximate optimal behavior without access to the true environment dynamics (Legg, 2008; Xing et al., 2025). Formally, the optimal policy under the world model f selects action sequences that maximize expected goal progress under simulated state transitions:

$$\pi^*_f(\hat{s}_t, g) = \underbrace{\arg\max_{a'_{t:T'-1}}}_{\text{possible actions}} \sum_{\hat{s}_{t+1:T'}} \underbrace{\left(\sum_{k=t}^{T'-1} \gamma_k r(\hat{s}_k, g) + \gamma_{T'} V^g_{\pi,f}(\hat{s}_{T'}) \right)}_{\text{goal progress}} \prod_{j=t}^{T'-1} \underbrace{p_f(\hat{s}_{j+1} | \hat{s}_j, a'_j)}_{\substack{\text{simulation with} \\ \text{world model}}}. \quad (1)$$

We refer to this form of deliberation as **simulative reasoning** (System II): the agent proposes candidate actions, predicts their consequences through the world model f, and selects the sequence that maximizes expected long-term progress. In contrast to black-box chain-of-thought, which expects planning capabilities to emerge from fitting training data, simulative reasoning provides a general-purpose planning mechanism grounded in verifiable next-state prediction, applicable across diverse tasks without domain-specific procedures. Xing et al. (2026) show that augmenting any baseline policy with a reasonably accurate world model yields a mixed policy that is no worse, and strictly better when simulative reasoning identifies an improvement.

In practice, exact optimization over Equation 1 is intractable. We denote by π_f a simulative planner that approximates π_f^*. Its output is a *plan* c_t encoding the current belief, a selected action sequence, and predicted future states:

$$c_t = (\hat{s}_t, a'_t, \hat{s}_{t+1}, a'_{t+1}, \ldots, \hat{s}_{T'}) \sim p_{\pi_f}(\cdot \mid \hat{s}_t). \tag{2}$$

The plan provides structured grounding for coherent behavior over long horizons: expected future states can be used to assess plan progress and detect violated expectations, while planned actions can guide execution if the predicted state is encountered later. Given a plan c_t, the agent selects concrete actions through an actor α that handles fine-grained reasoning and direct action: $a_t \sim p_\alpha(\cdot \mid \hat{s}_t, c_t)$. This reactive component captures execution patterns that are difficult to encode in structured plans, and enables fast response when deliberation is unnecessary.

2.2 From Unregulated Deliberation to Self-Regulation

In practice, the dominant approach to agent design does not construct explicit simulative plans c_t or regulate when planning occurs. Instead, the agent is implemented as a reactive policy π that generates a latent deliberation variable z_t before the action a_t, with planning expected to emerge implicitly:

$$p_\pi(a_t \mid \hat{s}_t) = \sum_{z_t} p_\pi(a_t \mid \hat{s}_t, z_t) \underbrace{p_\pi(z_t \mid \hat{s}_t)}_{\substack{\text{unregulated} \\ \text{deliberation}}}. \tag{3}$$

In current LLMs, z_t takes the form of chain-of-thought reasoning (OpenAI, 2024a; DeepSeek-AI, 2025); in vision-language-action models (VLAs), it may correspond to latent vectors (e.g., Helix (Figure AI, 2025)) or semantic action tokens (e.g., $\pi_{0.7}$ (Physical Intelligence et al., 2026)). In all cases, z_t carries no explicit beliefs about the current state, predicted future states, or contingency plans to ground action selection. Invoking simulative reasoning (System II) at every step makes these ingredients explicit (Deng et al., 2025; Wang et al., 2025b; Ye et al., 2026), but is costly when replanning is unnecessary (e.g., simple continuations).

A more flexible approach is to regulate planning itself. Inspired by human decision-making (§1; Kahneman, 2011), we introduce the *configurator* κ (System III) that explicitly governs the agent's planning behavior. The configurator outputs a decision u_t based on the current belief state $\hat{s}_t$, controlling whether and how planning occurs (e.g., whether to make a new plan, continue an existing one, or skip planning entirely). Separating the configurator κ (System III), simulative planner π_f (System II), and actor α (System I), the agent's action distribution decomposes into three stages:

$$p_\pi(a_t \mid \hat{s}_t) = \sum_{u_t, c_t} \underbrace{p_\alpha(a_t \mid \hat{s}_t, c_t)}_{\substack{\text{actor} \\ \text{(System I)}}} \underbrace{p_{\pi_f}(c_t \mid \hat{s}_t, u_t)}_{\substack{\text{simulative planner} \\ \text{(System II)}}} \underbrace{p_\kappa(u_t \mid \hat{s}_t)}_{\substack{\text{configurator} \\ \text{(System III)}}}. \tag{4}$$

The decomposition defines the variable production (regulation decisions u_t, structured plans c_t, and actions a_t) but doesn't prescribe how each component reasons internally; either the configurator or the planner may involve free-form reasoning as part of their output distribution (§3.2). Prior paradigms realize only subsets of Equation 4: effort-adaptive approaches (e.g., Lou et al., 2025; Wang et al., 2025a) learn u_t over modes of unstructured thought without simulative plans c_t; mode routing (e.g., Chen et al., 2025b) makes a single decision u_1 at task onset; and workflow distillation (e.g., Li et al., 2025c) routes among predefined modules without simulative plans c_t or a free-form actor α (§5).

3 Instantiating Self-Regulated Simulative Reasoning

We now implement and train the decomposition of §2 as SR^{2}AM, a family of agentic LLMs for interactive math, science, data analysis, and web information-seeking tasks. The LLM itself serves as

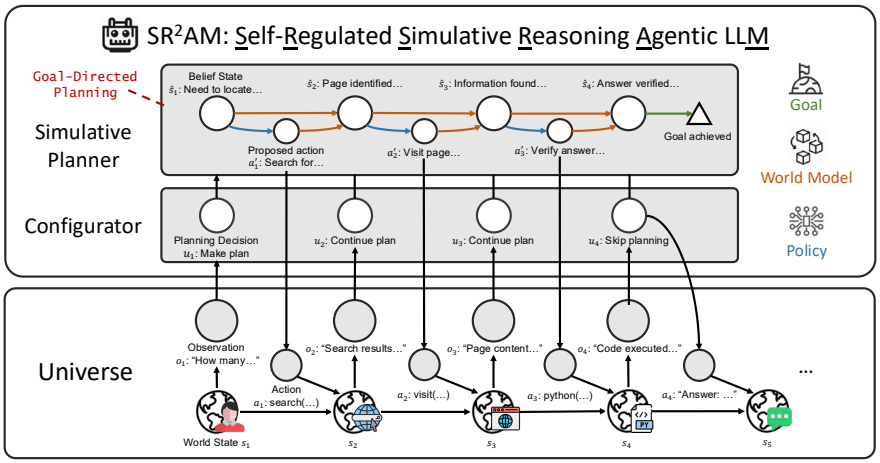

Figure 1: **Illustration of self-regulated simulative reasoning in SR^{2}AM**. At each step, the configurator (System III) decides whether to invoke, continue, or skip planning. When invoked, the simulative planner (System II) simulates proposed actions and predicted belief states for goal-directed planning. The actor (System I) handles fine-grained reasoning and direct action (not depicted). Both configurator and planner are realized as distinct stages in an LLM's chain-of-thought reasoning. The example depicts a web information-seeking task.

the world model in language space, with the configurator and planner realized as distinct stages of its chain-of-thought alongside free-form reasoning and acting (Figure 1).

3.1 Environment and Tools

At each time step t, the model receives observation o_t (consisting of prior reasoning context, actions, and tool outputs), forms a belief state $\hat{s}_t$, and selects an action a_t by calling one of several tools or generating a final text response. Following prior work (Jin et al., 2025; Li et al., 2025b; Cheng et al., 2025), we equip the agent with three tools: a web search engine (*web_search*), a web browser that crawls and summarizes page content given a visit goal (*visit_tool*), and a stateless Python sandbox for computation and data processing (*python_repl_tool*). The model can take up to $T_{\max}$ actions; at termination, reward $r(s_T, g)$ is computed based on the trajectory and final answer (§3.3). Full tool specifications and implementation details are provided in Appendix C.

3.2 Supervised Data Construction

The three-system decomposition is general and can be learned from scratch. To speed up learning with prior knowledge, we use pretrained LLMs to construct supervised data encoding configurator decisions (System III) and simulative plans (System II) alongside free-form reasoning (System I), via two approaches used for SR^{2}AM-v0.1 and v1.0, respectively.

Approach 1: Multi-Module Inference (v0.1) As a first approach demonstrating feasibility, we implement the configurator κ (System III) and planner π_f (System II) as separate prompted LLMs, augmented with additional LLM-based modules for belief formation (e.g., user intent interpretation, progress summarization, plan reflection, and free-form reasoning). These modules are supplied as callable tools for the configurator, which may invoke them freely before deciding on the next action: when further planning is necessary, it activates the relevant capabilities; when planning is complete, it selects an action to execute. The resulting traces are constructed by interleaving the configurator's thoughts with the output of each invoked module. Trajectories are filtered for answer correctness and minimum reasoning complexity. This approach is agnostic to the choice of LLM; for our main experiments, we use `o4-mini` (OpenAI, 2025e). Full collection details, including module selection per task type, retry logic, and prompts, are provided in Appendices D.1 and P.

Approach 2: Plan Reconstruction (v1.0) Our primary approach leverages `DeepSeek-V3.2` (DeepSeek, 2025), whose chain-of-thought traces contain useful information for both configurator decisions and task planning. We first collect interleaved thinking-acting

trajectories $(o_1, z_1, a_1, \ldots, o_T, z_T, a_T)$ from a pretrained LLM, then instruct an annotator LLM ψ to reconstruct configurator decisions (System III) and simulative plan content (System II) from these traces. For each step t, the annotator outputs a decision $\hat{u}_t \in \{0, 1\}$ for whether planning is necessary. If $\hat{u}_t = 1$, it infers a structured plan:

$$\hat{c}_t = (\hat{s}^c_t, a'_t, \hat{s}^c_{t+1}, a'_{t+1}, \ldots, \hat{s}^c_{T'}, a'_{T'}) \sim q_\psi(\cdot \mid o_1, z_1, a_1, \ldots, a_T, \hat{c}_{<t}),$$

where $\hat{s}^c_t$ summarizes conditions relevant to planning, $(a'_t, \ldots, a'_{T'})$ describe proposed actions, and $(\hat{s}^c_{t+1}, \ldots, \hat{s}^c_{T'})$ are predicted future states. One planning step in $[t, T']$ may summarize multiple real-time steps or a fraction of one, enabling hierarchical planning at multiple time scales. During inference, generating c_t amounts to the LLM jointly inferring the current state, proposing actions, and predicting their consequences, implicitly serving as encoder, policy, and world model within a single generation pass. The annotated plans are appended to the original model thoughts z_t, preserving the content of the original reasoning (System I) while augmenting it with structured plans (System II) that the configurator (System III) can selectively invoke. For web-browsing questions involving highly uncertain operations, we truncate plans to at most 2 steps. Collection and annotation details are provided in Appendix D.2.

3.3 Reinforcement-Learning-Based Refinement

After supervised finetuning, we train the models through RL to coordinate Systems I, II, and III for task success. For each task $g = (q, a^*)$, the agent generates configurator decisions u_t (System III), planner outputs c_t (System II), and actions a_t (System I), while the environment returns observations o_{t+1}, continuing for T steps until a final answer or $T_{\max}$ steps.

We define the reward as a combination of three binary signals: an answer reward r_{answer} measuring answer correctness via an LLM judge, a structure reward r_{struct} for format compliance across the trajectory, and a format reward r_{final} for final-answer extractability. These are combined into a piecewise function that prioritizes answer correctness while providing gradient signal for structural compliance even in unsuccessful trajectories (Appendix E.1). We optimize using an adapted version of Group Relative Policy Optimization (GRPO, Shao et al., 2024) with asymmetric clipping (Yu et al., 2025), sampling G trajectories per prompt and computing group-normalized advantages. For models of 30B and above, we filter truncated trajectories to prevent format collapse (Li et al., 2025a). The full RL objective derivation is provided in Appendix E.2.

3.4 Training Data and Hyperparameters

We build our training dataset from open-source math, science, tabular, and web reasoning datasets. For v0.1, we sample from Guru (Cheng et al., 2025) and multi-hop QA datasets (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Wu et al., 2025b), yielding 4,845 supervised examples after construction and filtering. For v1.0, we additionally incorporate MegaScience (Fan et al., 2025) and several web reasoning datasets (Wu et al., 2025a; Tao et al., 2025; Shi et al., 2025; Gao et al., 2025), yielding 10,787 supervised examples. For RL, we perform difficulty-based filtering (Cheng et al., 2025; Gao et al., 2025), retaining questions with intermediate Pass@K rates to ensure informative gradient signals. SR^{2}AM-v0.1-8B is trained from `Qwen3-8B` (Qwen Team, 2025g); SR^{2}AM-v1.0-30B from `Qwen3-30B-A3B-Thinking-2507` (Qwen Team, 2025f). Full dataset composition, filtering protocol, and training hyperparameters are provided in Appendix F.

4 Experiments

4.1 Experiment Setup

Evaluation Benchmarks We evaluate on 11 representative benchmarks across four categories: math (AIME-24 (Mathematical Association of America, 2024), AIME-25 (MAA Communications, 2024), MATH-500 (Hendrycks et al., 2021)), science (GPQA-Diamond (Rein et al., 2024), SuperGPQA (M-A-P Team et al., 2025), HLE (Phan et al., 2025)), tabular analysis (FinQA (Chen et al., 2021b), MultiHier (Zhao et al., 2022)), and web information seeking (BrowseComp (Wei et al., 2025), GAIA-103 (Mialon et al., 2023), XBench-DeepSearch (Chen et al., 2025a)). For HLE, we use the 500-question subset following Li et al. (2025c).

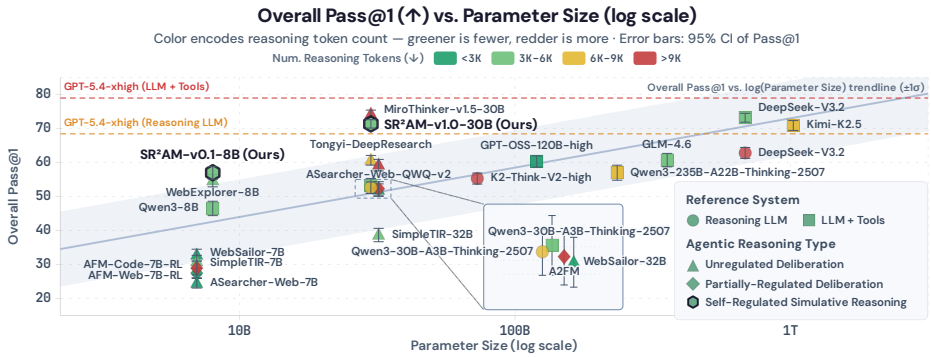

Figure 2: **Overall Pass@1 vs. Parameter Size (log scale).** Marker shape denotes system or reasoning type; color encodes reasoning token counts (greener = fewer, redder = more); trendline shows linear fit ($\pm 1\sigma$); dashed lines mark GPT-5.4-xhigh for reference. Both SR^{2}AM implementations perform well relative to their parameter sizes while using few reasoning tokens. Error bars show 95% CIs.

Baselines We compare against two types of agentic reasoning discussed in §2 that produced comparable models, and include reference systems to contextualize performance relative to pretrained LLMs. Full baseline details and inference configurations are described in Appendix G.

- **Reference Systems**: based on pretrained LLMs and not trained for agentic behavior. We evaluate *Reasoning LLMs* via direct prompting without tool use (GPT-5.4-xhigh (OpenAI, 2026), DeepSeek-V3.2 (DeepSeek, 2025), K2-Think-V2-high (K2 Think Team et al., 2026), and Qwen3-30B-A3B-Thinking-2507 (Qwen Team, 2025f)), and *LLM + Tools* which receive the same tool harness as our models (GPT-5.4-xhigh, Kimi-K2.5 (Moonshot AI, 2026), DeepSeek-V3.2, GLM-4.6 (Z.ai Team, 2025), GPT-OSS-120B-high (OpenAI, 2025d), Qwen3-8B (Qwen Team, 2025g), Qwen3-30B-A3B-Thinking-2507 (Qwen Team, 2025f), and Qwen3-235B-A22B-Thinking-2507 (Qwen Team, 2025d)).
- **Unregulated Deliberation**: agentic LLMs trained to reason and act with unconstrained reasoning (Tongyi-DeepResearch (Tongyi DeepResearch Team et al., 2025), MiroThinker-v1.5-30B (MiroMind Team et al., 2025), WebSailor-(7B/32B) (Li et al., 2025b), ASearcher-Web-(7B/QWQ-v2) (Gao et al., 2025), SimpleTIR-(7B/32B) (Xue et al., 2025), and WebExplorer-8B (Liu et al., 2025a))
- **Partially-Regulated Deliberation**: agentic LLMs realizing a subset of our proposed three-system decomposition (A^{2}FM (Chen et al., 2025b) for *Mode Routing* and AFM-(Web-7B/Code-7B) (Li et al., 2025c) for *Workflow Distillation*)

Evaluation Protocol and Metrics We report overall Pass@K (Chen et al., 2021a) following Li et al. (2025a), defined as the unweighted average of Pass@K across all M datasets (Pass@1 by default, Pass@3 where applicable). For reasoning efficiency, we report the average number of reasoning tokens per trajectory, defined as the total tokens generated by the agent excluding environment observations and tool outputs. Error bars are produced by stratified cluster bootstrap over questions with 10,000 replicates. Full evaluation settings (timeouts, context lengths, generation hyperparameters, test-set duplication for stability, and per-benchmark scoring functions) are provided in Appendix H. We provide sensitivity analysis on the choice of judge LLMs in Appendix A.4.

4.2 Main Results

We present our main results along two dimensions: overall Pass@1 averaged across all 11 benchmarks, and average reasoning tokens per trajectory (Figure 2); per-benchmark results are in Appendix L.

Task Performance. As Figure 2 shows, both SR^{2}AM implementations perform well relative to their parameter sizes. SR^{2}AM-v0.1-8B achieves the highest overall Pass@1 among 7–8B systems (57.0 vs. 54.7 for the next-best WebExplorer-8B), on par with Qwen3-235B-A22B-Thinking (57.0) in the same tool harness and within 4 points of certain 30–32B unregulated agentic LLMs and 120B–357B pretrained LLMs with tools. SR^{2}AM-v1.0-30B reaches 71.3, within 2 points of DeepSeek-V3.2

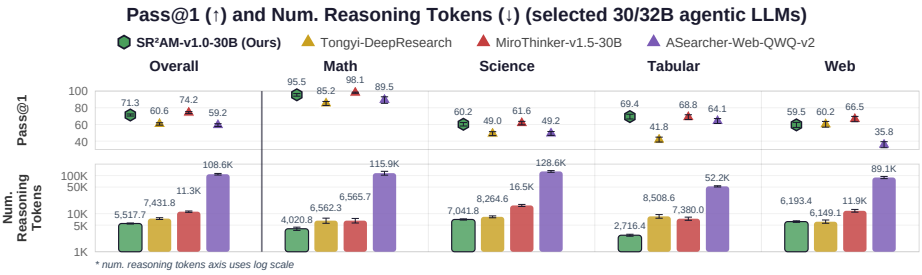

Figure 3: **Pass@1 (dots) and Average reasoning tokens (bars, log scale)** for four 30/32B agentic LLMs, broken down by task category and overall (Error bars show 95% CIs).

Outcome	Math	Science	Tabular	Web	Overall
Success	87.1±2.4 (14.4k)	77.1±1.3 (23.9k)	78.8±2.5 (4.9k)	72.8±2.1 (15.6k)	**78.6**±1.1 (58.8k)
Failure	54.7±9.1 (1.6k)	63.5±1.9 (19.7k)	69.0±3.8 (3.6k)	43.8±1.6 (38.2k)	**51.6**±1.4 (63.1k)
All	83.9±3.2 (16.0k)	70.9±1.3 (43.7k)	74.7±2.2 (8.5k)	52.2±1.6 (53.8k)	**64.6**±1.2 (121.9k)

Table 1: **Accuracy of world-model predictions** for SR^{2}AM-v1.0-30B: the percentage of plan steps whose expected outcome aligns with the subsequent trajectory, by task category and trajectory outcome (*All*: all trajectories; *Overall*: all plan steps pooled across categories). Values are shown with 95% CIs (bootstrap over tasks) and the number of plan steps in parentheses; alignment is judged by an LLM (Appendix S).

(685B, 73.2), on par with Kimi-K2.5 (1.0T, 70.9) in the same tool harness, and exceeding GPT-5.4-xhigh as a text-only reasoning LLM (68.4) despite falling short of the latter in the tool harness (78.9). Among 30–32B agentic LLMs, SR^{2}AM-v1.0-30B outperforms a wide range of baselines representing both unregulated and partially-regulated deliberation, and trails only MiroThinker-v1.5-30B (74.2), which consumes roughly twice as many reasoning tokens.

Reasoning Efficiency. Figure 3 details the comparison among 30–32B agentic LLMs. Among 7–8B models, SR^{2}AM-v0.1-8B consumes 3,698 reasoning tokens per trajectory on average, on par with the strongest 7–8B baselines (e.g., WebExplorer-8B at 3,617) while achieving higher Pass@1; systems using fewer tokens (601–2,609) trail by at least 24 points. Among stronger 30–32B agentic LLMs, SR^{2}AM-v1.0-30B consumes 25.8–94.9% fewer reasoning tokens while achieving higher accuracy than all but MiroThinker-v1.5-30B. In particular, compared to MiroThinker-v1.5-30B, SR^{2}AM-v1.0-30B is 2.9 points lower in Pass@1, but consumes 51.2% fewer reasoning tokens (5,518 vs. 11,295). These results suggest that self-regulated simulative planning achieves strong task success while controlling reasoning length, compared to both unregulated and partially regulated paradigms of similar scale. We analyze more dimensions of efficiency in Appendix A.3

4.3 Analysis of Self-Regulated Simulative Reasoning

We now examine whether the world model predictions are accurate, how RL shapes planning behavior, and whether the empirical gains above can be attributed to the System I+II+III decomposition, focusing on v1.0 (our primary contribution) with additional v0.1 analyses at the end. Unless otherwise noted, analyses use 30 randomly sampled examples from each of the 11 benchmarks (330 samples), repeated 3 times (990 runs).

Accuracy of World Model Predictions Simulative planning (System II) in SR^{2}AM-v1.0-30B grounds each plan in world-model predictions, (i.e., every plan step pairs a proposed action a'_t with a predicted future state $\hat{s}^c_{t+1}$). To test whether these predictions are accurate and whether their accuracy correlates with task success, we use an LLM judge to assess whether the expected outcome of each plan step aligns with the subsequent trajectory (Appendix S), and tabulate the results on the full test set trajectories by task outcome in Table 1. Overall, the world model is accurate on math, science, and tabular tasks (70.9–83.9% of plan steps aligned), which involve largely deterministic deduction and computation, but less so on web tasks (52.2%), likely due to the high uncertainty of web search and browsing (§3.2). Nevertheless, accuracy is clearly associated with success: it is higher in successful

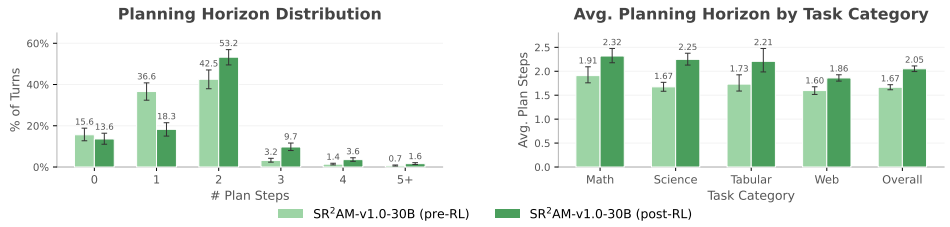

Figure 4: **Planning horizon analysis of SR^{2}AM-v1.0-30B before (light green) and after (green) RL. Left:** RL shifts mass toward longer plans (3+ steps: 5.3%→14.9%) while unplanned turns stay stable (15.6%→13.6%), deepening System II without increasing System III frequency. **Right:** Average plan horizon increases across all categories, with the largest gain in science (34.7%) and smallest in web (16.3%), consistent with web's shorter feasible horizon under environmental uncertainty (Error bars show 95% CIs over questions).

	SFT		SFT + RL	
Method	Pass@1	# Reasoning Tokens	Pass@1	# Reasoning Tokens
SR^{2}AM-v1.0-30B	66.6±1.5	4,925±166	**72.7***±0.7	5,672±60
− Structured Plan (System II)	65.2±1.6	4,602±140	71.0±0.7	4,945±54
− Selective Planning (System III)	65.2±1.6	5,451±183	68.6±0.7	5,779±68
Original Teacher CoT	65.3±1.5	3,844±129	71.8±0.7	4,555±55

Table 2: **Ablation on plan reconstruction for SR^{2}AM-v1.0-30B, before and after RL.** − *structured plan* replaces the plan's state-action-future-state structure with unstructured text; − *selective planning* disables the configurator's ability to skip planning. All variants receive the same SFT and RL recipe. Values are mean ± 95% CIs over 3 (SFT) and 18 (SFT+RL) evaluation runs on a fixed set of 330 questions; each checkpoint is evaluated shortly after its training stage, and variants are compared within each column. *Significantly higher than every other variant ($p < 0.05$, one-sided paired t-test over questions).

than in failed trajectories in every task category (72.8–87.1% vs. 43.8–69.0%), consistent with the hypothesis that simulative reasoning provides grounding for long-horizon behavior (§2.1). Testing whether this relationship is causal, by controlling world-model accuracy, is left to future work.

RL Refines Planning Horizon Without Overplanning We next ask whether RL incentivizes longer-horizon or simply more frequent planning, comparing planning-horizon distributions before and after RL (Figure 4, left). Results show RL clearly shifts mass toward longer-horizon plans while planning frequency stays stable, indicating that the configurator (System III) learns to plan more deeply when it plans, rather than more often, so RL channels improvement through planning quality (System II horizon) rather than quantity (System III frequency). This pattern holds across all four task categories (Figure 4, right). Web has the shortest post-RL horizon (1.86), consistent with its shorter feasible horizon under environmental uncertainty. Representative trajectories (Appendix M) show that simulative plans catch errors that unconstrained reasoning misses, RL produces more anticipatory plans with fallback strategies, and occasional over-planning on simple tasks leaves room to calibrate *when to stop* planning.

Component Ablation of System Decomposition We ablate two components of the SR^{2}AM-v1.0 SFT data: simulative planning (System II; c_t's state-action-future-state structure) and self-regulation (System III; the configurator's ability to set $u_t = 0$). Each variant is trained with the same SFT and RL recipe (§3) and compared against direct SFT on the original teacher CoT (Table 2). After SFT, all variants reach comparable accuracy. With RL training, every variant improves (by 3.4-6.5 points), while our proposed approach reaches the highest Pass@1 (72.7), significantly above every other variant. Removing selective planning (System III) matters most: it yields the smallest RL improvement (2.7 points below ours; one-sided $p < 0.05$) and the lowest final accuracy (68.6), despite producing the longest reasoning at the SFT stage (11% more tokens than ours). Conversely, direct SFT on teacher CoT produces the shortest reasoning (about 20% fewer tokens than ours at both stages) but lower final accuracy. Together, these results indicate that additional reasoning does not by itself improve accuracy; what matters is where and how it is spent, which is what the configurator regulates and the simulative planner structures.

Additional Analyses We summarize two analyses on v0.1, and present full details and figures in Appendix A. **To examine whether the advantage of self-regulated simulative reasoning persists through RL**, we compare RL training of SR^{2}AM-v0.1-8B against `Qwen3-8B` (same base model, unregulated deliberation) under identical settings. Results show unregulated deliberation dramatically increases reasoning tokens without a corresponding accuracy gain, while SR^{2}AM-v0.1-8B consistently uses fewer tokens and steadily converges to higher accuracy. **To disentangle the System I+II+III structure from simply having any supervised initialization**, we compare SFT data from our v0.1 pipeline against unregulated reasoning data from the same teacher LLM (`o4-mini`). Results show our structured data outperforms unregulated deliberation across data scales, with stronger teacher LLMs (e.g., `Qwen3-Coder-480B-A35B-Instruct` (Qwen Team, 2025b)) further improving results."

5 Related Work

We briefly discuss the landscape of agentic reasoning, and provide a more comprehensive discussion in Appendix B. Frontier reasoning models (OpenAI, 2024a; DeepSeek-AI, 2025; Yang et al., 2025) apply extended chain-of-thought uniformly, with no mechanism to modulate reasoning horizon or structure. The resulting inefficiency has motivated external regularization via length penalties (Arora & Zanette, 2025; Aggarwal & Welleck, 2025) or supervised compression (Kang et al., 2024; Xia et al., 2025), and user-specified effort controls (OpenAI, 2024b; Liu et al., 2025b), which impose constraints without modeling internal autonomy. Effort-adaptive approaches (Lou et al., 2025; Zhang et al., 2025b; Fang et al., 2025; Shen et al., 2025) train models to regulate the *amount* of reasoning, with recent extensions to agentic settings (Paglieri et al., 2026; Wang et al., 2025a; Jali et al., 2026), but operate along a single axis without constructing simulative plans (System II). Mode routing (Chen et al., 2025b; Jiang et al., 2025) makes a one-time decision at task onset without per-turn reassessment (System III). Workflow distillation (Li et al., 2025c) internalizes multi-agent orchestration (Wu et al., 2023; Hong et al., 2024; ByteDance, 2026; MiroMindAI, 2026) but inherits rigid sequencing and lacks free-form reasoning (System I) and variable-horizon planning (System II). World-model-based planning, from classical MPC (Camacho & Bordons, 2007) through learned latent world models (Schrittwieser et al., 2020; Hafner et al., 2019b) to visual simulators (Assran et al., 2025; Xiang et al., 2025; Ye et al., 2026), grounds action selection in predicted future states but invokes simulation obligatorily. LLM-based planning approaches (Yao et al., 2023a; Hao et al., 2023; Zhou et al., 2023; Deng et al., 2025) demonstrate that LLMs can serve as world models, yet their planning remains obligatory with no selective invocation (Qian et al., 2026).

6 Conclusion

We have argued and empirically verified that efficient agentic reasoning benefits from decomposing deliberation into simulative planning (System II) governed by learned self-regulation (System III), operating alongside reactive execution (System I). Across four diverse categories of interactive reasoning tasks, this decomposition yields competitive accuracy at substantially lower reasoning cost with a single simulative planning framework, supporting simulative reasoning as a general-purpose planning mechanism; ablation studies further confirm that removing either System II or System III lowers post-RL accuracy. RL with self-regulated simulative reasoning produces qualitatively different improvements than RL without it: longer-horizon plans rather than more frequent plans, and controlled growth in token consumption. These findings suggest that the need for ever-longer reasoning traces in current agentic models may be a consequence of building the agent as an undifferentiated reactive policy, which lacks genuine internal agency over its own planning. We discuss limitations and future directions in Appendix I.

7 AI-Use Disclosure Statement

In this work, we used generative AI tools for "generating synthetic datasets", "implementing methods", "cleaning and reformatting datasets", "supporting qualitative and thematic data analysis", "interpreting results", and "assisting with translation". Where LLMs serve as components of the research itself, namely synthetic trajectory generation, translation of non-English training questions, and LLM judges for rewards and evaluation, their use is documented in Appendices F and H. The intellectual

core of the work was developed by the authors without AI assistance: we did not use generative AI tools for "developing theoretical models or conceptual frameworks", "formulating mathematical claims", "proposing or refining hypotheses", or "designing and providing feedback on research methodology or experiments". As this paper contains no proofs, "providing critical ingredients for proving mathematical claims" and "assisting in the writing of proofs" are not applicable. In preparing the manuscript, we additionally used generative AI tools for "creating or modifying scientific figures", "drafting parts of the research paper", "editing the research paper to improve readability", "identifying relevant literature", "summarizing or analyzing existing literature", and "formatting references".

All AI-assisted work was reviewed by the authors before inclusion in the paper. We manually inspected random samples of the LLM-generated synthetic data before using it for training, and all AI-generated code was reviewed and tested by the authors. AI-assisted figures and qualitative analyses were checked against the underlying data, and every number in tables reformatted with AI assistance was cross-checked against the original experimental outputs. We treated AI-suggested interpretations as hypotheses rather than conclusions, verifying each interpretation reported in the paper through manual inspection of raw results and trajectories, consistency across experiments, and agreement with the formal framework in Section 2. Likewise, AI-drafted text was revised by the authors, and all AI-identified or AI-summarized references were checked against the original sources.

The authors take full responsibility for the content of this paper, including any text, claims, code, data, or figures produced with the aid of generative AI.

REFERENCES

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

Pranjal Aggarwal and Sean Welleck. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *arXiv preprint arXiv:2503.04697*, 2025.

Anthropic. Claude 3.7 Sonnet and Claude Code, February 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Daman Arora and Andrea Zanette. Training Language Models to Reason Efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Axolotl maintainers and contributors. Axolotl: Open Source LLM Post-Training. https://github.com/axolotl-ai-cloud/axolotl, May 2023. Software.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15791–15801, 2025.

Akhiad Bercovich, Itay Levy, Izik Golan, et al. Llama-Nemotron: Efficient Reasoning Models. *arXiv preprint arXiv:2505.00949*, 2025. URL https://arxiv.org/abs/2505.00949.

ByteDance. DeerFlow, 2026. URL https://github.com/bytedance/deer-flow/tree/main-1.x. Use the main-1.x branch for DeerFlow 1.x; accessed 2026-04-04.

Bytedance-Seed-Foundation-Code-Team, Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, Bowen Li, Linyi Li, Boyi Liu, Jiaheng Liu, Kaibo Liu, Qi Liu, Shukai Liu, Siyao Liu, Tianyi Liu, Tingkai Liu, Yongfei Liu, Rui Long, Jing Mai, Guanghan Ning, Z. Y. Peng, Kai Shen, Jiahao Su, Jing Su, Tao Sun, Yifan Sun, Yunzhe Tao, Guoyin Wang, Siwei Wang, Xuwu Wang, Yite Wang, Zihan Wang, Jinxiang Xia, Liang Xiang,

Xia Xiao, Yongsheng Xiao, Chenguang Xi, Shulin Xin, Jingjing Xu, Shikun Xu, Hongxia Yang, Jack Yang, Yingxiang Yang, Jianbo Yuan, Jun Zhang, Yufeng Zhang, Yuyu Zhang, Shen Zheng, He Zhu, and Ming Zhu. FullStack Bench: Evaluating LLMs as Full Stack Coders. *arXiv preprint arXiv:2412.00535v6*, 2025. URL https://arxiv.org/abs/2412.00535v6.

E. F. Camacho and C. Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2 edition, 2007. ISBN 978-0-85729-398-5. doi: 10.1007/978-0-85729-398-5.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, Kenkun Liu, Rui Wang, Run Li, Tong Niu, Wenlong Zhang, Wenqi Yan, Xuanzheng Wang, Yuchen Zhang, Yi-Hsin Hung, Yuan Jiang, Zexuan Liu, Zihan Yin, Zijian Ma, and Zhiwen Mo. xbench: Tracking Agents Productivity Scaling with Profession-Aligned Real-World Evaluations. *arXiv preprint arXiv:2506.13651*, 2025a. URL https://arxiv.org/abs/2506.13651.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Qianben Chen, Jingyi Cao, Jiayu Zhang, Tianrui Qin, et al. A^2FM: An Adaptive Agent Foundation Model for Tool-Aware Hybrid Reasoning. *arXiv preprint arXiv:2510.12838*, 2025b.

Xingyu Chen et al. Do Not Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021b. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Nilabjo Dey, Yonghao Zhuang, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Taylor W. Killian, Haonan Li, Mikhail Yurochkin, Eric P. Xing, Zhengzhong Liu, and Zhiting Hu. Revisiting Reinforcement Learning for LLM Reasoning from A Cross-Domain Perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/c2f71567cd53464161cab3336e8fc865-Abstract-Datasets_and_Benchmarks_Track.html.

DeepSeek. DeepSeek-V3.2: Efficient Reasoning & Agentic AI, December 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-V3.2.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

Mingkai Deng, Jinyu Hou, Zhiting Hu, and Eric Xing. General agentic planning through simulative reasoning with world models. *arXiv preprint arXiv:2507.23773*, 2025.

Xin Ding, Jianyu Wei, Yifan Yang, Shiqi Jiang, Qianxi Zhang, Hao Wu, Fucheng Jia, Liang Mi, Yuxuan Yan, Weijun Wang, Yunxin Liu, Zhibo Chen, and Ting Cao. AdaNav: Adaptive reasoning with uncertainty for vision-language navigation, 2025. URL https://arxiv.org/abs/2509.24387.

Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Y Zou. Cost-of-pass: An economic framework for evaluating language models. In *International Conference on Learning Representations*, volume 2026, pp. 34103–34127, 2026.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. MegaScience: Pushing the Frontiers of Post-Training Datasets for Science Reasoning. *arXiv preprint arXiv:2507.16812*, 2025. URL https://arxiv.org/abs/2507.16812.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM Learns When to Think. *arXiv preprint arXiv:2505.13379*, 2025.

Figure AI. Helix: A Vision-Language-Action Model for Generalist Humanoid Control. https://www.figure.ai/news/helix, 2025. Accessed: 2026-05-06.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond Ten Turns: Unlocking Long-Horizon Agentic Search with Large-Scale Asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025. URL https://arxiv.org/abs/2508.07976.

Qiyue Gao, Kun Zhou, Jiannan Xiang, Zihan Liu, Dequan Yang, Junrong Chen, Arif Ahmad, Cong Zeng, Ganesh Bannur, Xinqi Huang, et al. World Reasoning Arena. *arXiv preprint arXiv:2603.25887*, 2026.

Carlos E. Garcia, David M. Prett, and Manfred Morari. Model Predictive Control: Theory and Practice—A Survey. *Automatica*, 25(3):335–348, 1989. ISSN 0005-1098. doi: 10.1016/0005-1098(89)90002-2.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. Inverse Scaling in Test-Time Compute. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=NXgyHW1c7M.

Google. Try Deep Research and our new experimental model in Gemini, your AI assistant. https://blog.google/products-and-platforms/products/gemini/google-gemini-deep-research/, December 2024. Accessed: 2026-04-04.

David Ha and Jürgen Schmidhuber. World Models. *arXiv preprint arXiv:1803.10122*, 2018. URL https://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://arxiv.org/abs/2103.03874.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.00352.

Zhiting Hu and Tianmin Shu. Language models, agent models, and world models: The law for machine reasoning and planning. *arXiv preprint arXiv:2312.05230*, 2023.

Neharika Jali, Anupam Nayak, and Gauri Joshi. Not all turns are equally hard: Adaptive thinking budgets for efficient multi-turn reasoning in agents, 2026. URL https://arxiv.org/abs/2604.05164.

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.14631.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

K2 Think Team, Taylor W. Killian, Varad Pimpalkhute, Richard Fan, Haonan Li, Chengqian Gao, Ming Shan Hee, Xudong Han, John Maggs, Guowei He, Zhengzhong Liu, and Eric P. Xing. K2 Think V2: A Fully-Sovereign Reasoning Model, 2026. URL https://mbzuai.ac.ae/news-events/news/k2-think-v2-fully-sovereign-reasoning-model.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3oT: Generating Shorter Chain-of-Thought without Compromising Effectiveness. *arXiv preprint arXiv:2412.11664*, 2024. URL https://arxiv.org/abs/2412.11664.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

LangChain Inc. LangGraph, 2026. URL https://github.com/langchain-ai/langgraph. Open-source framework for building stateful agents; accessed 2026-04-04.

Shane Legg. *Machine Super Intelligence*. PhD thesis, Università della Svizzera italiana, June 2008. URL https://sonar.ch/global/documents/317954.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Dingchu Zhang, Xixi Wu, Jialong Wu, Xinyu Wang, Zile Qiao, Zhen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebSailor-V2: Bridging the Chasm to Proprietary Agents via Synthetic Data and Scalable Reinforcement Learning. *arXiv preprint arXiv:2509.13305*, 2025a. URL https://arxiv.org/abs/2509.13305.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. WebSailor: Navigating Super-human Reasoning for Web Agent. *arXiv preprint arXiv:2507.02592*, 2025b. URL https://arxiv.org/abs/2507.02592.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-Agents: End-to-End Agent Foundation Models via Multi-Agent Distillation and Agentic RL. *arXiv preprint arXiv:2508.13167*, 2025c.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, et al. Webexplorer: Explore and evolve for training long-horizon web agents. *arXiv preprint arXiv:2509.06501*, 2025a.

Zhengzhong Liu, Liping Tang, Linghao Jin, Haonan Li, Nikhil Ranjan, Desai Fan, Shaurya Rohatgi, Richard Fan, Omkar Pangarkar, Huijuan Wang, et al. K2-V2: A 360-Open, Reasoning-Enhanced LLM. *arXiv preprint arXiv:2512.06201*, 2025b.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective, 2025c. URL https://arxiv.org/abs/2503.20783.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. AdaCoT: Pareto-Optimal Adaptive Chain-of-Thought Triggering via Reinforcement Learning. *arXiv preprint arXiv:2505.11896*, 2025.

M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, et al. SuperGPQA: Scaling LLM Evaluation across 285 Graduate Disciplines. *arXiv preprint arXiv:2502.14739*, 2025. URL https://arxiv.org/abs/2502.14739.

MAA Communications. 2024-25 AIME Thresholds Are Available, December 2024. URL https://maa.org/news/aime-thresholds-are-available/. Updated January 6, 2025.

Mathematical Association of America. American Invitational Mathematics Examination (AIME), 2024. URL https://www.maa.org/math-competitions/american-invitational-mathematics-examination-aime.

John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 1955. URL https://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf. Proposal dated August 31, 1955.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A Benchmark for General AI Assistants. *arXiv preprint arXiv:2311.12983*, 2023.

MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, Wenhan Dou, Yue Deng, Yunjie Fu, Junqi Ge, Chenxia Han, Tammy Huang, Zhenhang Huang, Jerry Jiao, Shilei Jiang, Tianyu Jiao, Xiaoqi Jian, Lei Lei, Ruilin Li, Gen Luo, Tiantong Li, Xiang Lin, Ziyuan Liu, Zhiqi Li, Jie Ni, Qiang Ren, Pax Sun, Shiqian Su, Chenxin Tao, Bin Wang, Wenhai Wang, Haonan Wang, James Wang, Jin Wang, Jojo Wang, Letian Wang, Shizun Wang, Weizhi Wang, Zixuan Wang, Jinfan Xu, Sen Xing, Chenyu Yang, Hai Ye, Jiaheng Yu, Yue Yu, Muyan Zhong, Tianchen Zhao, Xizhou Zhu, Yanpeng Zhou, Yifan Zhang, and Zhi Zhu. MiroThinker: Pushing the Performance Boundaries of Open-Source Research Agents via Model, Context, and Interactive Scaling. *arXiv preprint arXiv:2511.11793*, 2025. URL https://arxiv.org/abs/2511.11793.

MiroMindAI. MiroFlow, 2026. URL https://github.com/MiroMindAI/MiroFlow. Open-source research-agent framework; accessed 2026-04-04.

Moonshot AI. Kimi K2.5: Visual Agentic Intelligence. https://www.kimi.com/blog/kimi-k2-5, 2026. Accessed: 2026-04-05.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-Training Elicits Concise Reasoning in Large Language Models. *arXiv preprint arXiv:2502.20122*, 2025. URL https://arxiv.org/abs/2502.20122.

Allen Newell, John C Shaw, and Herbert A Simon. Report on a General Problem-Solving Program. Technical Report P-1584, RAND Corporation, 1959. URL https://cs.uky.edu/~sgware/reading/papers/newell1959report.pdf.

OpenAI. Learning to Reason with LLMs, 2024a. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. OpenAI o1 and new tools for developers, December 2024b. URL https://openai.com/index/o1-and-new-tools-for-developers/.

OpenAI. Introducing Codex. https://openai.com/index/introducing-codex/, May 2025a. Accessed: 2026-04-04.

OpenAI. Computer-Using Agent, 2025b. URL https://openai.com/index/computer-using-agent/.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, April 2025c. Accessed: 2026-04-05.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, August 2025d. URL https://openai.com/index/gpt-oss-model-card/.

OpenAI. OpenAI o3 and o4-mini System Card. https://openai.com/index/o3-o4-mini-system-card/, April 2025e. Official system card for OpenAI o3 and o4-mini.

OpenAI. Introducing Deep Research, 2025f. URL https://openai.com/index/introducing-deep-research/.

OpenAI. Introducing GPT-5.4, March 2026. URL https://openai.com/index/introducing-gpt-5-4/.

OpenRouter. Qwen3 30B A3B Thinking 2507: API pricing and providers. https://openrouter.ai/qwen/qwen3-30b-a3b-thinking-2507, 2026. Accessed: September 25, 2026.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning When to Plan: Efficiently Allocating Test-Time Compute for LLM Agents, February 2026. URL https://arxiv.org/abs/2509.03581v3. Version 3, revised February 17, 2026; originally posted September 3, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, et al. Humanity's Last Exam. *arXiv preprint arXiv:2501.14249*, 2025.

Physical Intelligence. Physical Intelligence (π). https://www.pi.website/, 2024. Accessed: 2026-05-06.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Cheng Qian, Emre Can Acikgoz, Bingxuan Li, Xiusi Chen, Yuji Zhang, Bingxiang He, Qinyu Luo, Dilek Hakkani-Tür, Gokhan Tur, Yunzhu Li, et al. Current Agents Fail to Leverage World Model as Tool for Foresight. *arXiv preprint arXiv:2601.03905*, 2026.

Qwen Team. Qwen3-32B. https://huggingface.co/Qwen/Qwen3-32B, April 2025a. Model card.

Qwen Team. Qwen3-Coder-480B-A35B-Instruct. https://huggingface.co/Qwen/Qwen3-Coder-480B-A35B-Instruct, July 2025b. Model card.

Qwen Team. Qwen3-235B-A22B-Instruct-2507. https://huggingface.co/Qwen/Qwen3-235B-A22B-Instruct-2507, 2025c. Model card, accessed 2026-04-05.

Qwen Team. Qwen3-235B-A22B-Thinking-2507. https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507, 2025d. Model card, accessed 2026-04-05.

Qwen Team. Qwen3-30B-A3B-Instruct-2507. https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507, 2025e. Model card, accessed 2026-04-05.

Qwen Team. Qwen3-30B-A3B-Thinking-2507. https://huggingface.co/Qwen/Qwen3-30B-A3B-Thinking-2507, 2025f. Model card, accessed 2026-04-05.

Qwen Team. Qwen3-8B. https://huggingface.co/Qwen/Qwen3-8B, 2025g. Model card, accessed 2026-04-05.

Qwen Team. Qwen3-Next-80B-A3B-Instruct. https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct, 2025h. Model card, accessed 2026-04-05.

Qwen Team. Qwen3.6-27B: Flagship-Level Coding in a 27B Dense Model. https://qwen.ai/blog?id=qwen3.6-27b, April 2026. Model card: https://huggingface.co/Qwen/Qwen3.6-27B.

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. In *COLM*, 2024.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement Learning with Chain-of-Action-Thought Enhances LLM Reasoning via Autoregressive Search. *arXiv preprint arXiv:2502.02508*, 2025. URL https://arxiv.org/abs/2502.02508.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. TaskCraft: Automated Generation of Agentic Tasks. *arXiv preprint arXiv:2506.10055*, 2025. URL https://arxiv.org/abs/2506.10055.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961. URL https://www.nature.com/articles/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, December 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/10.1126/science.aar6404.

Peter Steinberger. Introducing OpenClaw. https://openclaw.ai/blog/introducing-openclaw, January 2026. OpenClaw Blog, accessed 2026-04-04.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between Underthinking and Overthinking: An Empirical Study of Reasoning Length and correctness in LLMs. *arXiv preprint arXiv:2505.00127*, 2025. URL https://arxiv.org/abs/2505.00127.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998. URL https://incompleteideas.net/book/first/the-book.html.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebShaper: Agentically Data Synthesizing via Information-Seeking Formalization. *arXiv preprint arXiv:2507.15061*, 2025. URL https://arxiv.org/abs/2507.15061.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Jiaqi Wang, Kevin Qinghong Lin, James Cheng, and Mike Zheng Shou. Think or Not? Selective Reasoning via Reinforcement Learning for Vision-Language Models. *arXiv preprint arXiv:2505.16854*, 2025a.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, et al. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *arXiv preprint arXiv:2510.16907*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, 2022.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebDancer: Towards Autonomous Information Seeking Agency. *arXiv preprint arXiv:2505.22648*, 2025a. URL https://arxiv.org/abs/2505.22648.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in Web Traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10290–10305, 2025b. doi: 10.18653/v1/2025.acl-long.508. URL https://aclanthology.org/2025.acl-long.508/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

Siye Wu, Jian Xie, Yikai Zhang, Aili Chen, Kai Zhang, Yu Su, and Yanghua Xiao. ARM: Adaptive Reasoning Model. *arXiv preprint arXiv:2505.20258*, 2025c. URL https://arxiv.org/abs/2505.20258.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.165. URL https://aclanthology.org/2025.emnlp-main.165/.

Jiannan Xiang, Yi Gu, Zihan Liu, Zeyu Feng, Qiyue Gao, Yiyan Hu, Benhao Huang, Guangyi Liu, Yichi Yang, Kun Zhou, et al. Pan: A world model for general, interactable, and long-horizon world simulation. *arXiv preprint arXiv:2511.09057v1*, 2025. URL https://arxiv.org/abs/2511.09057v1.

Eric Xing, Mingkai Deng, Jinyu Hou, and Zhiting Hu. Critiques of world models. *arXiv preprint arXiv:2507.05169v1*, 2025. URL https://arxiv.org/abs/2507.05169v1.

Eric Xing, Mingkai Deng, and Jinyu Hou. Critique of Agent Model. *arXiv preprint arXiv:2606.23991*, 2026. URL https://arxiv.org/abs/2606.23991.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-End Reinforcement Learning for Multi-Turn Tool-Integrated Reasoning. *arXiv preprint arXiv:2509.02479*, 2025. URL https://arxiv.org/abs/2509.02479.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin

Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023b.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying Long Chain-of-Thought Reasoning in LLMs. *arXiv preprint arXiv:2502.03373*, 2025. URL https://arxiv.org/abs/2502.03373.

Qiying Yu et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Z.ai Team. GLM-4.6. https://huggingface.co/zai-org/GLM-4.6, 2025. Model card, accessed 2026-04-05.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills, October 2025a. Accessed: 2026-04-04.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. AdaptThink: Reasoning Models Can Learn When to Think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3716–3730, 2025b.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6588–6600, 2022. doi: 10.18653/v1/2022.acl-long.454. URL https://aclanthology.org/2022.acl-long.454/.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT:A Scalable lightWeight Infrastructure for Fine-Tuning. *arXiv preprint arXiv:2408.05517*, 2024. URL https://arxiv.org/abs/2408.05517.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group Sequence Policy Optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

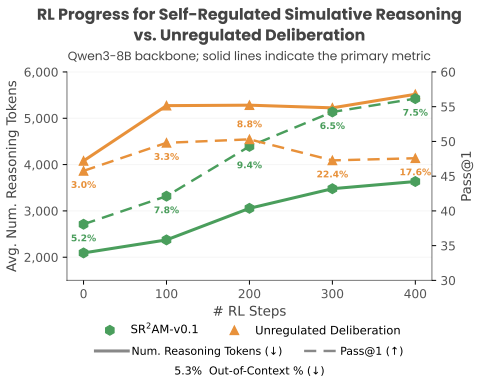

Figure 5: RL training progress over 400 steps from self-regulated simulative reasoning (v0.1, green) vs. unregulated deliberation (orange). Solid lines show average reasoning tokens per trajectory (left axis, ↓ better); dashed lines show overall Pass@1 (right axis, ↑ better); percentages indicate out-of-context rate (↓ better) at each checkpoint. Unregulated deliberation increases token consumption with diminishing accuracy returns, while self-regulated simulative reasoning channels improvement through planning quality with controlled token growth.

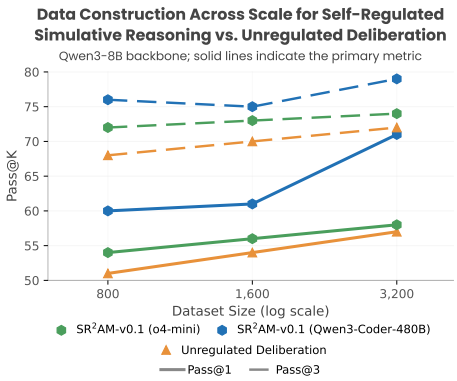

Figure 6: Pass@1 (solid) and Pass@3 (dashed) on validation set after SFT on `Qwen3-8B` with increasing data sizes (800–3200 examples). Three data construction approaches are compared: unregulated reasoning reconstruction (using `o4-mini` actions with `GPT-4.1`-inferred reasoning, orange), our SFT data construction pipeline (v0.1) using `o4-mini` (green), and the same pipeline using `Qwen3-Coder-480B-A35B-Instruct` (blue). With the same teacher LLM, self-regulated simulative reasoning outperforms unregulated deliberation at every data scale (by 1.2–3.5 points in Pass@1 and 1.5–3.4 points in Pass@3), confirming the value of the structured decomposition itself.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL Scaling. `https://github.com/THUDM/slime`, 2025. GitHub repository. Corresponding author: Xin Lv.

A Additional Analyses

A.1 The Advantage of Self-Regulated Simulative Reasoning Persists Through RL

To examine whether the advantage of self-regulated simulative reasoning persists through RL training, we compare the RL training of SR^{2}AM-v0.1-8B against Qwen3-8B, the same base model performing unregulated deliberation, for 400 steps under identical settings except for max completion tokens, which is increased to 16,384 for better test performance. We report pass rate, reasoning tokens per trajectory, and out-of-context rate at every 100 steps. The same principle is also reflected in v1.0's training dynamics (Table 2, first row), where RL lifts Pass@1 from 66.6 to 72.7 with only moderate token growth (4,925 to 5,672).

Results (Figure 5) show that unregulated deliberation leads to dramatic increases in reasoning token consumption without corresponding accuracy gains: token consumption starts high ($\sim$4,100) and grows steadily, but pass rate peaks at step 200 and subsequently declines, accompanied by rising context overflow (22.4%). In contrast, SR^{2}AM-v0.1-8B consistently uses fewer reasoning tokens ($\sim$2,100 $\rightarrow$ 3,600) while steadily converging to higher accuracy with negligible context overflow. At step 400, it uses 34.1% fewer reasoning tokens than the unregulated model while achieving higher pass rate (56.2 vs 47.6). This confirms that the advantage of the System I+II+III decomposition not only persists but amplifies through RL: improvement flows through planning quality (System II) and regulation decisions (System III) rather than reasoning volume, yielding better convergence within practical context budgets.

System	# Calls		Generated (k)			Total LLM Tokens (k)	Cost-per-success ($)
	LLM	Tool	Reason.	Action	Total		
SR^{2}AM-v1.0-30B	**18.68**	**17.75**	**5.52**	**1.26**	**6.78**	**28.79**	**0.029647**
Tongyi-DeepResearch	25.58	18.74	7.43	5.19	12.62	40.35	0.059177
MiroThinker-v1.5-30B	51.45	49.97	11.30	4.54	15.84	76.28	0.067918
ASearcher-Web-QWQ-v2	45.50	27.27	108.58	9.15	117.73	125.46	0.172070

Table 3: **End-to-end token accounting and cost-per-success** for 30–32B agentic LLMs on the full test set (k = thousands of tokens; bold = lowest in each column). Values are means over all 8,219 test trajectories.

A.2 Disentangling the System I+II+III Structure from Supervised Initialization

To disentangle the contribution of our proposed System I+II+III structure from simply having any supervised initialization, we compare SFT data constructed through our multi-module inference pipeline (v0.1) against data encoding unregulated reasoning collected from the same teacher LLM. To collect the latter, we follow WebSailor (Li et al., 2025b) and use `o4-mini` to collect action traces without reasoning content, and `gpt-4.1` to infer the missing free-form reasoning content one step at a time. We fine-tune `Qwen3-8B` on progressively larger subsets (800 to 3,200 examples) from each pipeline, with all other hyperparameters fixed. To further test the impact of the LLM powering data collection, we additionally run our multi-module pipeline with `Qwen3-Coder-480B-A35B-Instruct`, a strong open-weight instruct LLM. We evaluate on a held-out validation set of 300 examples drawn from the same distribution as our SFT training data, averaged over 6 runs to reduce variance.

With `o4-mini` as the teacher LLM for both pipelines, data with our proposed structure outperforms unregulated reasoning reconstruction at every data scale (1.2–3.5% in Pass@1 and 1.5–3.4% in Pass@3, Figure 6). Since both pipelines use the same underlying LLM, this improvement can be attributed to the structured System I+II+III content rather than the quality of the underlying LLM. Replacing `o4-mini` with `Qwen3-Coder-480B-A35B-Instruct` leads to further improvement, particularly at larger scales: Pass@1 increases from 60.9 to 72.9 as data grows from 1,600 to 3,200 examples. This indicates that the benefit of the structured decomposition scales with both data quantity and teacher quality.

A.3 Additional Efficiency Analysis

Whereas §4.1 focuses on reasoning efficiency and counts each model's reasoning token cost, we provide here a more detailed analysis of inference cost for the four 30–32B agentic LLMs in Figure 3. For each trajectory, we count the number of LLM and tool calls, generated tokens (including reasoning tokens, action tokens for tool calls and the final answer) and total LLM tokens (including tool outputs). Following A^{2}FM (Chen et al., 2025b) and the cost-of-pass framework Erol et al. (2026), we also report the expected inference cost per correct answer, defined as below:

$$\mathrm{CpS} = (P_{\text{in}} T_{\text{in}} + P_{\text{out}} T_{\text{out}})/R,$$

Where T_{in} and T_{out} are the per-task prefill and generated tokens, R is per-task accuracy, and P_{in}, P_{out} are the list prices of each system's base model: \$0.20 / \$2.40 per million input / output tokens for `Qwen3-30B-A3B` (SR^{2}AM, Tongyi-DeepResearch and MiroThinker; OpenRouter, 2026) and \$0.28 / \$0.84 for Qwen2.5-32B-Instruct (ASearcher-Web's QwQ-32B; Chen et al., 2025b). As shown in Table 3, SR^{2}AM-v1.0-30B costs the fewest in total tokens and cost-per-success.

A.4 Sensitivity to the Choice of Judge LLMs

Our evaluation uses LLM APIs following existing work (Table 5). To test whether the metrics are sensitive to this choice, we re-judge one full evaluation run of SR^{2}AM-v1.0-30B (8,219 attempts across all 11 benchmarks) with two open-weight judges, `Qwen3-235B-A22B-Instruct-2507` and `DeepSeek-V3.2`. Because both LLMs are sensitive to the exact formatting of numerical answers (e.g., units, percentages, and rounding), we add format-robust instructions to the judge prompts for FinQA, MultiHier, and GAIA-103. Table 4 reports agreement with the original judges, Cohen's κ, and the resulting Pass@1 by task type.

		Qwen3-235B		DeepSeek-V3.2		Pass@1		
Task type	n	Agr. (%)	κ	Agr. (%)	κ	Original	Qwen	DeepSeek
Math	2,420	100.0±0.0	1.00±0.00	100.0±0.0	1.00±0.00	95.5±1.6	95.5±1.6	95.5±1.6
Science	2,292	99.0±0.4	0.98±0.01	97.9±0.6	0.96±0.01	60.2±2.1	59.4±2.1	58.5±2.1
Tabular	1,429	93.1±1.3	0.83±0.03	82.3±2.0	0.63±0.04	69.4±2.8	69.4±2.8	56.0±3.0
Web	2,078	97.8±0.9	0.96±0.02	97.7±0.8	0.95±0.02	59.5±3.0	57.9±3.1	58.5±3.0
Overall	8,219	98.0±0.3	0.95±0.01	95.8±0.4	0.90±0.01	71.3±1.2	70.7±1.2	68.2±1.2

Table 4: **Agreement of open-weight judges with the original judges, by task type,** on one evaluation run of SR^{2}AM-v1.0-30B (n = attempts, including test-set duplication). Agreement and Cohen's κ are pooled over attempts; Pass@1 is the unweighted mean over benchmarks, as in the main results. The lower tabular scores under `DeepSeek-V3.2` stem from over-strict numerical formatting (see text). Values represent 95% CI over questions

Both judges agree closely with the original judges: overall agreement is 98.0% ($\kappa = 0.95$) for `Qwen3-235B` and 95.8% ($\kappa = 0.90$) for `DeepSeek-V3.2`. Disagreements concentrate on tabular questions, whose answers are mostly numerical. There, `Qwen3-235B`'s disagreements run in both directions and leave Pass@1 unchanged, whereas `DeepSeek-V3.2` lowers the tabular Pass@1 by 13.4 points. To identify which judge is correct, we manually audited 60 randomly sampled attempts accepted only by the original judges: 92% are numerically equivalent to the reference answer and differ only in format or rounding (e.g., "11.83%" vs. 0.118), 3% reflect errors in the reference answer, and 5% (95% CI 2–14%) are genuine mismatches. The tabular gap therefore largely reflects `DeepSeek-V3.2`'s over-strictness on numerical formatting rather than errors by the original judges. Notably, `Qwen3-235B` also serves as the RL reward judge for SR^{2}AM-v1.0 (Appendix F); its high agreement with the independent evaluation judges indicates that the reward signal is consistent with how we evaluate.

B Extended Related Work

B.1 Unregulated Deliberation and External Controls

Unregulated Deliberation. Frontier reasoning models such as OpenAI o1/o3 (OpenAI, 2024a), DeepSeek-R1 (DeepSeek-AI, 2025), and Qwen3 (Yang et al., 2025) have demonstrated the power of extended chain-of-thought reasoning for complex tasks. While these models show some emergent sensitivity to task difficulty (DeepSeek-AI, 2025), there is no explicit mechanism to modulate the presence, horizon, or structure of reasoning, and reasoning length tends to grow rapidly during RL training without commensurate accuracy gains. The resulting inefficiency has motivated a line of work on *external* regularization, such as RL with length-penalized objectives (Arora & Zanette, 2025; Aggarwal & Welleck, 2025; Yeo et al., 2025), length preference optimization (Chen et al., 2024), or supervised compression of verbose chains into more concise forms (Kang et al., 2024; Munkhbat et al., 2025; Xia et al., 2025). These methods reduce reasoning cost on average, but the regularization is imposed externally (via reward shaping or distillation targets) rather than being learned as a situational decision by the model itself. In terms of the System I+II+III decomposition, these approaches operate entirely within System I (reactive execution with adaptive computation), with no explicit System II (simulative planning) or System III (learned self-regulation).

User-Specified Effort. Several frontier systems have introduced coarse-grained, user-facing controls over reasoning effort. Qwen3 (Yang et al., 2025) implements a think/no-think switching mechanism via chat templates, while o1/o3 (OpenAI, 2024b) and K2-V2 series (Liu et al., 2025b) allow configurable reasoning effort tiers (e.g., low/medium/high). These developments recognize that uniform reasoning is inefficient, but the regulation remains externally specified: the model itself has no autonomy to adjust its reasoning effort based on task demands (System III), and cannot refine a user-specified tier on a per-step basis as the interaction unfolds.

B.2 Partially Regulated Deliberation

Effort-Adaptive Approaches. A growing body of work trains models to regulate reasoning effort autonomously based on situational signals. Several works learn explicit switching policies via RL (Lou et al., 2025; Zhang et al., 2025b; Fang et al., 2025), compare rewards of thinking and non-thinking responses to the same input (Jiang et al., 2025), or select among coarse output formats (Wu et al., 2025c). Satori (Shen et al., 2025) introduces meta-action tokens for within-trace search control in math QA. More recent work extends from single-turn reasoning to multi-turn agentic settings: Paglieri et al. (2026) train a monolithic LLM to decide whether to invoke planning at each step in long-horizon environments; TON (Wang et al., 2025a) extend selective think-or-not reasoning to vision-language and agentic tasks via thought-dropout SFT followed by GRPO; and Jali et al. (2026) learn to allocate reasoning budgets across turns based on interaction history. AdaNav (Ding et al., 2025) learns uncertainty-guided reasoning triggers for vision-language navigation. These approaches share our motivation for situational regulation and realize a form of System III (deciding whether and how much to reason), but operate along a single axis without distinguishing among qualitatively different reasoning operations or constructing explicit simulative plans (System II).

Mode Routing. A^2FM (Chen et al., 2025b) goes beyond effort control by routing queries among qualitatively different execution regimes: instant response, internal reasoning, and tool-based agentic execution. This recognize that the *kind* of reasoning matters, not just its amount, realizing a coarse form of System III. However, the routing decision is typically made once at task onset and applied uniformly thereafter, without per-turn reassessment as the interaction unfolds, and without explicit simulative planning (System II) within any mode.

Externally Orchestrated Workflows. A natural approach to organizing complex agentic behavior is to decompose it into specialized modules and coordinate them through explicit control logic. AutoGen (Wu et al., 2023) and MetaGPT (Hong et al., 2024) define role-specialized agents (e.g., planner, coder, reviewer) and explicit interaction protocols governing their coordination. LangGraph (LangChain Inc., 2026) provides a graph-based orchestration framework in which developers define nodes, transitions, and control flow as a directed workflow. DeerFlow (ByteDance, 2026) and MiroFlow (MiroMindAI, 2026) use controller tiers and workflow templates to sequence reasoning steps such as planning, tool use, and reflection. These systems recognize the need for structured planning and regulation, implementing analogs of System II (planning stages) and System III (control flow governing when to plan, reflect, or act). However, the deliberation policy is prescribed by developers rather than learned by the agent, making the structure external to the model. The resulting behavior is governed by human-authored logic and is difficult to adapt to the specific demands of novel situations the agent encounters.

Workflow Distillation. The Chain-of-Agents (CoA, Li et al., 2025c) framework and its resulting Agent Foundation Model (AFM) take a step toward internalizing workflow-based structure by distilling multi-agent workflows into a single model that routes among predefined capabilities (e.g., planning, reflection, tool use). This eliminates external workflow orchestration at inference time, realizing a form of per-turn System III regulation. However, the model inherits the rigid sequencing constraints from its distillation templates (e.g., a plan must precede an action, reflection follows execution), leaving limited room for free-form reasoning that captures fine-grained patterns difficult to specify categorically (System I). The planning component also lacks the simulative, variable-horizon structure described in our formalization (System II): plans are action directives rather than state-action-future-state sequences grounded in predicted consequences.

B.3 Simulative Planning Without Self-Regulation

World-Model-Based Planning. Planning by simulating future states has a long history in sequential decision-making. Model predictive control (MPC) uses an explicit dynamics model to optimize action sequences over a receding horizon (Garcia et al., 1989; Camacho & Bordons, 2007). Model-based RL extends this idea by *learning* the dynamics model from interaction data, as pioneered by Dyna (Sutton, 1991) and later scaled through learned latent world models (Ha & Schmidhuber, 2018; Schrittwieser et al., 2020; Hafner et al., 2019b;a). More recently, world models for planning have been most extensively developed in visual and embodied domains, where agents simulate future states (e.g.,

latent embeddings, images, or video) to evaluate candidate actions before execution (Assran et al., 2025; Zhou et al., 2024; Xiang et al., 2025; Bar et al., 2025; Ye et al., 2026). Critiques of World Models (Xing et al., 2025) articulate a broader vision for this paradigm, where an agent leverages a world model as a sandbox to precompute possible world states and best responses for use during decision time. World Reasoning Arena (Gao et al., 2026) instantiate this paradigm by using a VLM to propose actions, a world model to simulate their consequences, and the same VLM again to select the best action accordingly. These approaches realize System II (simulative planning grounded in predicted state transitions) but invoke simulation obligatorily at every decision point, with no mechanism for the agent to skip or curtail planning when the context does not warrant it (System III). Qian et al. (2026) empirically demonstrate this limitation, finding that current agents struggle to decide *when* to invoke simulation, frequently misusing or failing to leverage world models even when they are available as tools.

LLM as World Model. LLMs trained on text that encodes sequential world structure inherit formal similarities to world models, possessing implied knowledge of state transitions and action consequences (Hu & Shu, 2023). Several works have explored leveraging this capacity for planning. Tree of Thoughts (Yao et al., 2023a) and LATS (Zhou et al., 2023) use an LLM to evaluate intermediate reasoning and action states, implicitly leveraging the model's predictive capacity to score candidates and guide search. RAP (Hao et al., 2023) makes this role explicit by using an LLM to simulate state transitions within a Monte Carlo Tree Search framework. SiRA (Deng et al., 2025) further extends this paradigm to complex environments by inferring and predicting over a natural-language-based latent state, demonstrating substantial improvements in long-horizon web browsing tasks. These methods demonstrate that LLMs can serve as effective world models for planning in language space (System II). However, they typically operate as *prompting pipelines*, where capabilities such as policy, world model, and goal evaluation are stitched together through rule-based orchestration, rather than being internalized as learned capabilities within a single model. VAGEN (Wang et al., 2025b) moves toward an integrated model that proposes an action and simulates the future state using natural language, but planning is still obligatory, and simulation horizon remains shallow and fixed. In all cases, the question of *when* to invoke simulative planning (System III) remains unaddressed.

C Environment and Tool Details

We follow the typical setup for interactive reasoning (Jin et al., 2025; Li et al., 2025b): at the beginning of the task, the model receives goal $g = (q, a^*)$ consisting of question q and, during training, a reference answer a^*. The first observation o_1 contains the question q (with answer format requirements where appropriate). At each following time step t, the model receives observation o_t consisting of previous reasoning context, actions, and tool outputs, and embeds them using the LLM to form the belief state $\hat{s}_t$. The model may take action a_t by calling one of several tools or generating a text response which ends the task. When called, the tool returns its output as part of the next observation o_{t+1}. The model can take up to $T_{\max}$ actions. At the end of the task, the reward $r(s_T, g)$ is computed based on the complete trajectory and final answer, as detailed in Appendix E. Following previous work, we provide the model with the tools described below (with their names as exposed to the model in parentheses):

Search Engine (*web_search*) A search engine for querying web-scale information beyond what is reliably stored within model parameters. We adopt the implementation of Li et al. (2025b), which takes inputs such as query, date, location, and number of results, and allows multiple simultaneous queries. For provider, we use both SerpAPI and Serper.Dev[1], which we find to be largely interchangeable through our experiments.

Web Browser (*visit_tool*) A web browser that crawls and summarizes webpage content given a system-provided visit goal. We adopt the implementation from Li et al. (2025b), which crawls the content of a website and summarizes it using an instruct LLM given the visit goal. To reduce processing latency and prevent out-of-context errors, we truncate webpage content to 28,000 tokens. We find that the choice of summarizer LLM makes little difference, and uniformly use

[1] `https://serpapi.com`, `https://serper.dev`

`Qwen3-30B-A3B-Instruct-2507` (Qwen Team, 2025e) for its balance of quality and efficiency.

Code Sandbox (*python_repl_tool*) A stateless Python sandbox for computation, numerical simulation, and data processing. We adopt SandboxFusion (Bytedance-Seed-Foundation-Code-Team et al., 2025) following Cheng et al. (2025), which runs self-contained Python scripts and returns printed outputs or errors. We pre-install common Python libraries to facilitate model usage. While a stateful sandbox may lead to more stable processing (as we find LLMs often assume the sandbox is persistent), our training does enable the models to use the stateless sandbox effectively for task execution.[2]

D Supervised Data Collection Details

D.1 Multi-Module Inference Details (v0.1)

For collection using `o4-mini`, we allow up to $T_{\max} = 30$ actions, and retry up to 3 times until the model returns a final answer. We then filter trajectories for answer correctness; to preserve sufficient reasoning behavior, only trajectories calling more than 3 reasoning modules or actions combined are kept. Through validation experiments, we find that mixing different sets of reasoning capabilities for different types of tasks can improve performance. For instance, we drop the summary module for tabular tasks and additionally drop the user intent module for web tasks, possibly due to the simpler structure of these types of questions.

For web tasks, we additionally use an instruct LLM (i.e., `GPT-4.1` (OpenAI, 2025c)) to enrich the configurator's reasoning with more details prior to action selection, which we find improves performance in practice. We also explore using `Qwen3-Coder-480B-A35B-Instruct` (Qwen Team, 2025b), an open-source instruct LLM, for the same pipeline, with the comparison provided in §A.2. A trajectory example, the full collection routine, and the collection prompts are described in Appendices N, O, and P, respectively.

D.2 Plan Reconstruction Details (v1.0)

Because the plan is a structured sequence rather than an unstructured block of text, the planning horizon $T' - t$ is explicitly defined and controllable by adjusting plan annotations in the training data for different task categories. During collection, we use `DeepSeek-V3.2` (DeepSeek, 2025) in thinking mode with 128K context and $T_{\max} = 100$ steps, retrying up to 5 times until the model returns a correct answer. Trajectories are filtered for answer correctness. Plan annotation is performed jointly over each trajectory in a single pass using `DeepSeek-V3.2` with thinking mode enabled. A trajectory example and the annotation prompt are provided in Appendices N and Q, respectively.

E RL Objective and Training Details

E.1 Reward Function

For each task $g = (q, a^*)$ from dataset $\mathcal{D}$, the agent π with parameters θ generates a trajectory:

$$\tau = (u_1, c_1, a_1, \ldots, o_T, u_T, c_T, a_T) \sim p_\mu^\pi(\cdot \mid o_1, \theta),$$

where p_μ^π denotes the joint distribution induced by the agent's policy π and the environment's transition dynamics μ.

Following DeepSeek-AI (2025) and Jin et al. (2025), the reward $r(\tau, g)$ combines three binary signals into a piecewise function that prioritizes answer correctness while providing a gradient signal for

[2] During inference and RL training, all tools are implemented as fully asynchronous to avoid synchronous execution bottlenecks during model rollouts.

structural compliance even in unsuccessful trajectories:

$$r(\tau, g) = \begin{cases} 1, & \text{if } r_{\text{answer}} \text{ and } r_{\text{struct}}, \\ 0.8, & \text{if } r_{\text{answer}} \text{ and } \neg r_{\text{struct}}, \\ 0.2, & \text{if } \neg r_{\text{answer}} \text{ and } r_{\text{struct}}, \\ 0.1, & \text{if } \neg r_{\text{answer}}, \ \neg r_{\text{struct}}, \text{ and } r_{\text{final}}, \\ 0, & \text{o.w.} \end{cases}$$

These reward values were fixed throughout all experiments and were not tuned.

E.2 GRPO Objective

To update the model, we use an adapted version of Group Relative Policy Optimization (GRPO, Shao et al., 2024). For each initial observation $o_1 \sim \mathcal{D}$, we sample a group of G trajectories $\{\tau_i\}_{i=1}^G$ from the current policy parameters θ_{old} and compute the corresponding rewards $\{r_i\}_{i=1}^G$. Let τ_{it}^π denote the t-th agent-generated token in trajectory i (excluding environment observations), and let $|\tau_i^\pi|$ be the total number of such tokens. We estimate the advantage $\hat{A}_{it}$ of each token via group-level z-score normalization, and compute the likelihood ratio $r_{it}(\theta)$ between the current and old parameters:

$$\hat{A}_{it} = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}, \qquad r_{it}(\theta) = \frac{p_\pi(\tau_{it}^\pi \mid o_1, \tau_{i,<t}, \theta)}{p_\pi(\tau_{it}^\pi \mid o_1, \tau_{i,<t}, \theta_{\text{old}})}.$$

Our training objective maximizes the clipped surrogate advantage, with asymmetric clipping bounds ϵ_{low} and ϵ_{high} (Yu et al., 2025):

$$\mathcal{J}(\theta) = \mathbb{E}_{o_1 \sim \mathcal{D}, \{\tau_i\}_{i=1}^G \sim p_\mu^\pi(\cdot|o_1, \theta_{\text{old}})} \left\{ \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|\tau_i^\pi|} \sum_{t=1}^{|\tau_i^\pi|} \min \left[r_{it}(\theta)\hat{A}_{it}, \text{clip}(r_{it}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\hat{A}_{it} \right] \right\}. \tag{5}$$

E.3 Additional Training Considerations

To stabilize training, we keep all model updates on-policy. For models of 30B and above, we follow Li et al. (2025a) and filter out trajectories that terminate due to truncation, as noisy training signal from malformed negative trajectories can cause format collapse. We experimented with several other techniques (e.g., dynamic sampling and GSPO (Zheng et al., 2025)), but did not observe better performance, so we keep the original GRPO objective, which delivers strong performance in practice. Recent work has proposed additional refinements such as token-level loss aggregation (Yu et al., 2025) and mean-only group normalization (Liu et al., 2025c), which may further stabilize training; we leave their integration to future work.

F Training Dataset Composition and Hyperparameters

F.1 Dataset and Data Construction

We build our training dataset from open-source math, science, tabular, and web reasoning datasets.

SR^{2}AM-v0.1 We use Guru (Cheng et al., 2025) for math, science, and tabular data (152,660 examples from 5 sources before filtering), and combine HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and WebWalkerQA-silver (Wu et al., 2025b) for web data (288,173 examples). To standardize the language, we translate Chinese questions from WebWalkerQA-silver into English using `Qwen3-32B` (Qwen Team, 2025a), while instructing the model to include Chinese proper names in parentheses. From the combined 440,833 examples, we sample 6,400 for self-regulation data construction, balanced proportionally to the square root of each domain's frequency. After construction and filtering, 4,845 supervised examples (20.7M tokens) remain.

SR^{2}AM-v1.0 We build on the v0.1 dataset and additionally include MegaScience (Fan et al., 2025) for math and science data, and WebDancer (Wu et al., 2025a), WebShaper (Tao et al., 2025), TaskCraft (Shi et al., 2025), and ASearcherLRM35k (Gao et al., 2025) for web data. As MegaScience is an automatically constructed dataset with possibly noisy responses, we only keep questions with answers fewer than 1,024 characters long. For TaskCraft, to ensure sufficient difficulty and answer reachability, we only keep questions with `valid_hop` at least 3 and exclude those requiring image tools. Due to the large size of MegaScience (1.25M), we sample only the same number as our new web examples combined (42,863 examples). After deduplication, the additional datasets amount to 86,087 examples. We sample 6,400 questions from this new set and combine them with the 6,400 from v0.1, for a total of 12,800 questions for self-regulation data construction, yielding 10,787 supervised examples (39.5M tokens) after filtering. For RL, we combine the new datasets with half of the v0.1 dataset to ensure a diverse distribution of data sources.

F.2 RL Data Filtering

GRPO-style objectives benefit from questions that produce diverse reward signals across sampled trajectories, as questions with uniform rewards yield zero advantages and uninformative gradients. While approaches like DAPO (Yu et al., 2025) address this through dynamic sampling at training time, the oversample-and-filter approach can reduce training efficiency. Instead, we follow previous approaches (Cheng et al., 2025; Gao et al., 2025) and perform difficulty-based filtering prior to training: we roll out each question K times using the pre-RL checkpoint and retain only those with Pass@$K \in [a, b]$. In practice, we do not necessarily need to run each question for the full K times to determine whether Pass@K falls within the target range (e.g., to ensure Pass@$K > 0$, it is enough to see one success), which improves filtering efficiency. For tasks where agentic rollouts are expensive (e.g., web-search-heavy reasoning), we follow Li et al. (2025c) and use a strong instruct LLM to sample N answers for each question, keeping questions with Pass@$N < c$, which reflects question difficulty based on parametric knowledge alone.

SR^{2}AM-v0.1 We take $K = 16$, $a = \frac{1}{16}$, and $b = \frac{15}{16}$. For web questions, we use `Qwen3-Next-80B-A3B-Instruct` (Qwen Team, 2025h) for filtering with $N = 32$ and $c = 0.3$. After filtering, we are left with 110,086 non-web questions and 124,596 web questions. Due to the large size, which may bias training, we downsample the web dataset to 33.3% of its original size, resulting in a final RL data mix of 151,618 examples.

SR^{2}AM-v1.0 We find the pre-RL checkpoint to be stronger, and thus impose more stringent requirements by taking $K = 8$, $a = \frac{1}{8}$, and $b = \frac{6}{8}$. For web questions, we use `Qwen3-235B-A22B-Instruct-2507` (Qwen Team, 2025c) with $N = 32$ and $c = 0.3$ for filtering. After filtering half of the dataset (except for v0.1 web data which we sample $\frac{1}{24}$ from), we are left with 7,795 v0.1 non-web questions, 5,154 v1.0 non-web questions, 4,555 v0.1 web questions, and 5,154 v1.0 web questions. To build the RL dataset, we take the filtered non-web data, mix in v0.1 web examples equal to 33.3% of the v0.1 non-web examples, and v1.0 web examples equal to the number of v1.0 non-web examples. The final mix contains 20,701 examples in total. During RL training, we filter the remaining parts of the dataset using intermediate checkpoints to keep data up-to-date with the model's capability.

F.3 Training Hyperparameters

SR^{2}AM-v0.1 We start with `Qwen3-8B` (Qwen Team, 2025g) as the base model. SFT is performed using Axolotl (Axolotl maintainers and contributors, 2023), a FSDP-based LLM finetuning library, with global batch size 32 and max context length 32,768, using Adam (lr=2.5e-5, linear warmup with ratio 0.1, cosine decay to 2.5e-6, weight decay 0.01) (Kingma & Ba, 2015) for 4 epochs following Bercovich et al. (2025). RL is performed using Slime (Zhu et al., 2025) with rollout batch size 128, 8 samples per prompt, max response length 8,192, max context length 40,960, temperature 0.8, max 40 action steps, $\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$, and Adam (constant lr=1e-6, weight decay 0.1). We train for 400 steps (51.2K examples) and stop when improvement slows down. For the reward function, we use the LLM judges from Guru (Cheng et al., 2025) for math and science questions, and the GAIA and SimpleQA judges from MiroThinker (MiroMind Team et al., 2025) for tabular and web questions, respectively. All LLM judges are implemented using `Qwen3-30B-A3B-Instruct-2507`. Using

8 and 32 Hopper GPUs respectively, SFT takes 3 hours and RL takes about 3.3 days. We also trained another model with `Qwen3-32B` as the base model, using the same hyperparameters except training for 300 steps instead of 400. However, the resulting model shows only modest improvement over the 8B variant, suggesting that the v0.1 data construction approach rather than model scale was the primary bottleneck. This observation motivated the v1.0 plan reconstruction approach described above.

SR^{2}AM-v1.0 We use `Qwen3-30B-A3B-Thinking-2507` (Qwen Team, 2025f) as a strong base model that supports long context natively. SFT is performed using ms-swift (Zhao et al., 2024), a Megatron-based LLM finetuning library, with global batch size 32, max context length 131,072, sample packing, and MoE auxiliary loss 0.01, using Adam (lr=1e-5, warmup fraction 0.05, cosine decay to 1e-6, weight decay 0.01) for 4 epochs. RL uses the same GRPO configuration as v0.1, except with max response length 16,384, max context length 122,880, temperature 1.0, top-p 0.95, and max 100 tool steps. We train for 200 steps (25.6K examples). Specifically, we first train for 160 steps, and then filter additional data using the intermediate checkpoint to obtain data of sufficient difficulty for the remaining 40 steps. We use the same reward function as v0.1, except using `Qwen3-235B-A22B-Instruct-2507` as a stronger LLM judge. Using 32 Hopper GPUs, SFT takes 1.5 hours and RL takes about 6 days.

For both implementations, we still observe reward and evaluation improvement by the end of training, indicating the models may be undertrained. Scaling training data and compute with more efficient fully asynchronous training remains an exciting direction for future work.

G Baseline Details

We organize baselines into three categories based on how they produce behavior, and describe the rationale for each below. For models with configurable reasoning effort (i.e., GPT-5.4, GPT-OSS-120B, and K2-Think-V2), we choose the highest tier (xhigh, high, and high, respectively).

G.1 Reference Systems

These systems are not trained for agentic behavior and serve as reference points for what pretrained LLMs can achieve. *Reasoning LLMs* are evaluated through direct prompting without tool use, representing a degenerate agent that must generate an answer in one step without environment interaction. To allow full exploration of the reasoning space, we do not impose any maximum completion token limit. *Pretrained LLMs with Tools* receive the same tool harness as our trained models but no agentic training, isolating the contribution of tool access on top of pretrained reasoning. These models use the same tools and inference settings described in §3.1.

G.2 Unregulated Deliberation

These agentic LLMs are trained to reason and act in tool-augmented environments, but rely on unconstrained chain-of-thought with no mechanism to control the presence, horizon, or structure of planning. Planning behavior, if any, must emerge implicitly from end-to-end training.

G.3 Partially-Regulated Deliberation

These systems introduce some form of adaptive reasoning but realize only a subset of the full System I+II+III decomposition. *Mode Routing* (A^{2}FM) performs a single routing decision at task onset, selecting among qualitatively different execution regimes (e.g., instant response, internal reasoning, or tool-based execution), without per-turn regulation or explicit simulative planning. *Workflow Distillation* (AFM) internalizes rule-based routing among predefined capabilities through distillation from multi-agent workflows, providing per-turn regulation but without free-form reasoning (System I) or simulative planning (System II).

G.4 Inference Settings

For all agentic LLM baselines, we use their default inference parameters and routines. For models that require a browsing summarization LLM, we use the same model described in §3.1.

H Evaluation Details

H.1 Evaluation Protocol

Following Li et al. (2025a), we report results on all benchmarks using consistent inference settings to ensure stability and reproducibility. To ensure reasonable inference time, we impose timeouts per turn at 10 minutes, per tool at 5 minutes, and per overall response at 60 minutes. For all models running through our tool harness (SR^2AM and pretrained LLMs with tools), we set the per-turn max completion tokens to 16,384. For our models, we use the RL rollout temperature (0.8 for SR^2AM-v0.1-8B, 1.0 for SR^2AM-v1.0-30B) and top-p=0.95; for pretrained LLMs with tools and reasoning LLM baselines, we use their default generation hyperparameters. For SR^2AM-v0.1-8B and Qwen3-8B, we set the max context length to 40,960 and the max turn limit to 50. For all other models, we set the max context length to 131,072 and the max turn limit to 100.

To stabilize evaluation on benchmarks with small sample sizes, we follow Chen et al. (2025b) and duplicate the test set 32× for AIME-24 and AIME-25, and 4× for GPQA-Diamond, GAIA-103, and XBench-DeepSearch; reported metrics are averaged across all duplicated runs. For HLE, we use the same 500-question subset as in Li et al. (2025c). All other benchmarks use their full test sets.

H.2 Evaluation Metrics

Each system is evaluated by its overall Pass@K (Chen et al., 2021a), defined as the unweighted average of Pass@K across all datasets. Given M datasets with N_i examples for the i-th dataset, and with $p_{ij}^K \in \{0, 1\}$ denoting the correctness of the j-th example in the i-th dataset over K passes:

$$\text{Overall Pass@}K = 100 \times \frac{1}{M} \sum_{i=1}^{M} \frac{1}{N_i} \sum_{j=1}^{N_i} p_{ij}^K$$

The evaluation function for each benchmark, including the judge LLM and scoring method, is summarized in Table 5. To facilitate answer extraction, we follow Cheng et al. (2025) and append a one-sentence instruction to all questions for all evaluated systems, asking the model to present its final answer wrapped in `\boxed{}`.

For reasoning efficiency, we report the average number of reasoning tokens per trajectory, defined as the total tokens generated by the agent (e.g., free-form reasoning, configurator decisions, and plans) excluding environment observations, actions, and tool outputs.

I Limitations and Future Work

We studied our hypothesis in the setting of language-based interactive reasoning across four task categories; extending the evaluation to embodied and multi-agent settings, where state representations are richer and coordination constraints arise, is a natural next step. We adopted LLMs as a simplified world model in language space, which may have limited predictive power in the wider physical and social worlds; integrating multimodal world models capable of next-state prediction over perceptual inputs would test the decomposition under more complex dynamics. Our evaluation primarily assessed self-regulated simulative reasoning in terms of its holistic contribution to task accuracy and reasoning efficiency, which are the metrics that matter most for the empirical challenge we identify. A complementary analysis of the configurator (e.g., alignment of its planning decisions with oracle hindsight) could further pinpoint where the decomposition has the most room to improve. On the engineering side, our models retain full interaction history, which could be improved with context management techniques such as retaining only recent tool responses; train on publicly available datasets, which could be augmented with more targeted synthetic data for challenging domains like web reasoning; and adopt standard tool implementations, which could benefit from richer interfaces such as full terminal access or Agent Skills.

Natural extensions beyond the current evaluation setting include grounding simulative reasoning in richer world models over multimodal state representations, extending self-regulation to govern the agent's own learning process beyond planning alone, and ultimately integrating hierarchical goal decomposition and evolving self-models toward agent models capable of organizing their own behavior, achieving the autonomy and adaptability typically associated with natural agents.

J ASSET LICENSES

Base Models. All Qwen3 models used in this work (Qwen Team, 2025g;f) are released under Apache 2.0.

Training Data. 2WikiMultihopQA (Ho et al., 2020), WebWalkerQA (Wu et al., 2025b), and ASearcherLRM35k (Gao et al., 2025) are released under Apache 2.0, and WebDancer (Wu et al., 2025a) and WebShaper (Tao et al., 2025) are released through the WebAgent repository under Apache 2.0. GURU (Cheng et al., 2025) and TaskCraft (Shi et al., 2025) are released under MIT. HotpotQA (Yang et al., 2018) is released under CC BY-SA 4.0. MuSiQue (Trivedi et al., 2022) is released under CC BY 4.0. MegaScience (Fan et al., 2025) is released under CC BY-NC-SA 4.0.

Evaluation Benchmarks. GPQA-Diamond (Rein et al., 2024) is released under CC BY 4.0, and SuperGPQA (M-A-P Team et al., 2025) under ODC-BY. HLE (Phan et al., 2025), FinQA (Chen et al., 2021b), MultiHiertt (Zhao et al., 2022), BrowseComp (Wei et al., 2025), and XBench-DeepSearch (Chen et al., 2025a) are released under MIT. GAIA (Mialon et al., 2023) does not declare a license and is distributed through a gated repository whose terms prohibit resharing; the maintainers of GPQA, BrowseComp, and HLE likewise ask users not to reveal examples publicly. MATH-500 is a subset of MATH (Hendrycks et al., 2021); both MATH and PRM800K, which defines the subset, are released under MIT. AIME (Mathematical Association of America, 2024) problems are publicly available competition problems.

Training Frameworks and Tools. Axolotl (Axolotl maintainers and contributors, 2023), ms-swift (Zhao et al., 2024), Slime (Zhu et al., 2025), and SandboxFusion (Bytedance-Seed-Foundation-Code-Team et al., 2025) are released under Apache 2.0.

K Evaluation Function Sources

Benchmark	Judge LLM	Scoring Method	Source
AIME-24	gpt-4.1-mini	Rule-based + LLM fallback	LLM360/Reasoning360[1]
AIME-25	gpt-4.1-mini	Rule-based + LLM fallback	LLM360/Reasoning360[1]
MATH-500	gpt-4.1-mini	Rule-based + LLM fallback	LLM360/Reasoning360[2]
GPQA-Diamond	gpt-4.1-mini	LLM judge	LLM360/Reasoning360[2]
SuperGPQA	gpt-4.1-mini	LLM judge	LLM360/Reasoning360[2]
FinQA	gpt-4.1	LLM judge	MiroMindAI/MiroThinker[3]
MultiHier	gpt-4.1	LLM judge	MiroMindAI/MiroThinker[3]
GAIA-103	gpt-4.1	LLM judge	MiroMindAI/MiroThinker[3]
BrowseComp	gpt-4.1	LLM judge	MiroMindAI/MiroThinker[4]
HLE	o4-mini	LLM judge (structured)	MiroMindAI/MiroThinker[5]
XBench-DeepSearch	gpt-4.1	LLM judge	MiroMindAI/MiroThinker[6]

[1] Rule-based grading uses functions from PRIME-RL/PRIME (via verl), derived from hendrycks/math, Microsoft/ToRA, and OpenAI/prm800k.
[2] Adapted from TIGER-AI-Lab/General-Reasoner
[3] Adapted from the GAIA scorer from WebSailor (Alibaba-NLP/WebAgent).
[4] Adapted by WebSailor from the official evaluation function at openai/simple-evals.
[5] Adapted from the official evaluation function at centerforaisafety/hle
[6] Adapted from the official evaluation function at xbench-ai/xbench-evals

Table 5: Sources of evaluation functions used across all benchmarks.

L Quantitative Evaluation Result by Benchmarks

Model	Size	# Reas. Tokens	Overall	Math			Science		Tabular		Web		
				AIME 24	AIME 25	MATH 500	GPQA-D	SuperGPQA	HLE	FinQA	MultiHier	BrowseComp	GAIA	XBench
Reference Systems														
Reasoning LLMs														
Qwen3-30B-A3B[a]	30B	8269.1±563	52.6±1.7	91.4±8.7	86.9±10.0	98.6±1.0	70.2±5.6	63.1±3.0	10.2±2.7	64.2±2.8	58.9±5.4	0.5±0.4	18.9±6.7	16.0±6.4
K2-Think-V2-high	73B	31905.2±1819	55.3±1.6	93.9±6.8	87.6±10.0	98.8±0.9	73.0±5.3	58.0±3.1	11.6±2.9	74.5±2.7	65.2±5.1	2.0±0.8	27.4±7.9	16.2±6.1
DeepSeek-V3.2	685B	9306.2±670	62.8±1.6	95.4±5.6	93.8±7.4	99.2±0.7	86.5±4.0	71.8±2.8	24.0±3.8	72.2±2.7	66.7±5.1	6.4±1.4	37.1±8.5	37.8±8.4
GPT-5.4-xhigh	—[b]	8810.3±514	68.4±1.5	96.4±4.9	99.4±0.5	99.8±0.3	92.2±3.3	73.5±2.7	40.8±4.3	76.5±2.5	69.0±4.9	14.8±1.9	42.0±8.7	48.2±8.5
Pretrained LLMs + Tools														
Qwen3-8B	8B	3932.5±166	46.5±2.1	62.3±11.7	46.8±14.3	94.4±2.1	53.4±5.4	49.8±3.2	3.4±1.6	70.6±2.7	58.9±5.2	0.9±0.5	26.0±6.8	44.8±7.4
Qwen3-30B-A3B[a]	30B	5410.1±397	53.1±2.2	73.9±13.0	66.7±14.0	97.2±1.4	67.9±5.6	62.4±3.0	9.8±2.7	67.1±2.8	64.9±5.1	1.7±0.7	36.9±8.0	36.0±7.6
GPT-OSS-120B-high	120B	2700.4±150	60.3±1.6	79.9±9.0	80.2±7.2	97.2±1.5	69.8±5.2	58.6±3.0	20.4±3.6	72.3±2.7	67.3±5.1	14.8±2.0	54.9±7.2	48.0±7.6
Qwen3-235B[c]	235B	6467.4±313	57.0±2.2	75.5±12.5	63.1±14.9	98.4±1.1	77.4±5.2	66.3±2.9	12.0±2.8	67.2±2.7	62.5±5.2	4.3±1.1	50.0±8.4	50.2±7.7
GLM-4.6	357B	3580.4±198	60.7±2.0	71.7±11.0	64.4±12.3	96.4±1.6	75.8±4.5	67.0±2.9	19.0±3.4	73.1±2.6	66.1±5.1	22.8±2.3	60.9±7.4	50.5±7.6
DeepSeek-V3.2	685B	3011.5±137	73.2±1.3	95.1±2.8	90.7±4.9	99.0±0.9	87.2±3.3	75.0±2.7	39.6±4.3	72.0±2.7	68.2±4.9	38.1±2.7	81.3±5.7	58.5±7.8
Kimi-K2.5	1024B	6413.1±322	70.9±1.4	89.6±4.3	81.6±7.7	98.4±1.1	83.6±3.7	72.5±2.8	34.4±4.2	71.6±2.7	67.3±5.1	35.3±2.6	73.5±6.8	72.0±6.9
GPT-5.4-xhigh	—[b]	4821.8±307	78.9±1.3	95.2±5.1	94.7±4.3	99.2±0.7	92.3±2.9	77.7±2.5	49.0±4.5	76.9±2.5	70.2±4.9	48.7±2.7	83.5±6.7	80.8±6.5
Unregulated Deliberation														
ASearcher-Web-7B	7B	601.4±53	24.5±1.4	12.9±8.0	6.2±4.6	60.8±4.3	33.1±4.5	35.7±2.9	6.8±2.3	45.9±3.0	27.1±4.8	1.6±0.7	20.4±6.1	18.5±5.6
SimpleTIR-7B	7B	3551.7±165	30.9±1.8	33.3±12.7	21.1±10.3	85.6±3.1	39.3±5.1	41.7±3.0	4.6±1.8	55.4±3.0	42.3±5.2	0.6±0.4	8.7±4.2	6.8±3.6
WebSailor-7B	7B	2211.5±55	32.8±1.7	60.6±8.7	53.0±10.2	55.8±4.4	24.6±4.3	28.0±2.8	9.4±2.6	18.5±2.3	7.4±2.8	4.7±1.2	35.2±7.4	63.0±7.6
WebExplorer-8B	8B	3616.9±212	54.7±1.8	79.5±8.5	66.6±10.8	93.2±2.2	58.0±5.4	60.3±3.0	14.8±3.1	65.8±2.8	45.8±5.4	9.4±1.6	43.4±7.8	64.5±7.5
SimpleTIR-32B	32B	5174.8±360	38.6±2.0	49.2±13.2	32.6±13.1	91.2±2.5	53.0±5.4	50.5±3.1	3.6±1.7	64.7±2.8	60.7±5.2	0.6±0.4	11.9±4.9	7.0±3.5
WebSailor-32B	32B	1055.2±20	51.8±1.8	72.9±9.0	80.3±9.7	77.6±3.6	47.6±5.4	47.3±3.1	12.2±2.9	62.4±2.9	48.5±5.4	6.9±1.4	46.1±8.4	68.5±7.6
ASearcher-Web-QWQ-v2	32B	108580.7±5120	59.2±1.7	92.2±5.6	79.3±10.2	97.0±1.5	66.3±5.4	65.0±3.0	16.4±3.2	67.5±2.7	60.7±5.4	7.1±1.4	50.5±7.9	49.8±7.4
Tongyi-DeepResearch	30B	7431.8±402	60.6±1.4	89.2±4.3	92.0±4.3	74.6±3.9	60.6±5.1	54.9±3.0	31.6±4.0	45.8±3.0	37.8±5.1	36.9±2.7	67.7±7.2	76.0±6.6
MiroThinker-v1.5	30B	11295.2±460	74.2±1.2	98.9±0.8	97.2±1.4	98.2±1.1	81.2±4.0	73.7±2.8	30.0±4.0	72.7±2.6	64.9±5.1	38.6±2.7	84.5±5.3	76.5±7.2
Partially-Regulated Deliberation														
Workflow Distillation														
AFM-Web-7B-RL	7B	2608.6±93	27.5±1.4	24.8±7.0	13.5±5.7	62.8±4.2	30.8±4.0	32.4±2.9	7.8±2.3	53.7±3.0	31.2±4.8	1.2±0.6	22.6±5.7	21.2±5.6
AFM-Code-7B-RL	7B	11205.5±539	28.9±2.0	39.7±14.1	30.1±13.9	83.0±3.3	17.0±4.0	23.6±2.6	2.8±1.5	65.1±2.8	44.9±5.4	0.0±0.0	7.8±4.7	4.0±3.1
Mode Routing														
A2FM	32B	23424.7±1698	52.3±2.1	70.3±11.9	62.6±13.1	94.6±2.0	57.6±5.2	55.3±3.1	10.3±2.7	70.3±2.7	54.3±5.3	7.6±1.5	38.8±6.9	53.2±7.6
Self-Regulated Simulative Reasoning (Ours)														
SR^{2}AM-v0.1-8B	8B	3697.6±150	57.0±1.8	83.8±9.7	74.2±10.3	96.2±1.7	60.9±5.3	58.0±3.0	14.2±3.1	73.6±2.6	64.6±5.1	4.8±1.2	46.1±7.6	50.5±7.8
SR^{2}AM-v1.0-30B	30B	5517.7±171	71.3±1.2	96.2±2.1	91.1±4.1	99.2±0.7	77.3±4.0	73.0±2.8	30.4±4.0	73.4±2.6	65.5±5.1	22.5±2.3	76.0±6.2	80.0±6.1

[a] `Qwen3-30B-A3B-Thinking-2507`.
[b] Parameter count is not publicly disclosed.
[c] `Qwen3-235B-A22B-Thinking-2507`.

Table 6: Per-benchmark results across reference systems and agentic reasoning paradigms on the final test set. Systems are grouped by paradigm and ordered by parameter size within each group, with ties broken by overall performance in ascending order and GPT-5.4 listed last where applicable. "# Reas. Tokens" reports the average number of reasoning tokens per problem. ± values are half-widths of 95% confidence intervals from a stratified cluster bootstrap (questions resampled with replacement within each benchmark, all repetitions of a question kept together; 10,000 replicates).

M Comparison of Self-Regulated Simulative Reasoning Across Models

Following the benchmark maintainers' requests, ground-truth answers in the examples are redacted ([redacted]).

Example 1: Web Research (GAIA)

In this example, all three models navigate the USGS NAS database and identify the same raw data (6 Nile + 1 Morelet's), but differ in what the configurator (System III) does next. Both **SR^{2}AM-v1.0-30B (SFT)** and **Unregulated CoT (SFT)** generate plans specifying only "provide final answer", a reactive, single-step plan that accepts the direct tally of 7. **SR^{2}AM-v1.0-30B** generates a plan (System II) that flags the need to "ensure no other species are present and verify," leading it to cross-validate against an authoritative source confirming 6. This illustrates the shift from reactive to anticipatory planning that RL encourages (§4.3): the planner encodes a verification step *before* committing.

Question

How many nonindigenous crocodiles were found in Florida from the year 2000 through 2020? You can get the data from the USGS Nonindigenous Aquatic Species database. Provide your final answer in the format `\boxed{YOUR_ANSWER}`.

SR^{2}AM-v1.0-30B (Ours) — 45 messages ✓ **Correct**

Assistant: I need to find the number of nonindigenous crocodiles in Florida from 2000 to 2020. I should search for data from the USGS NAS database.

Plan :

- **Current State:** Need to locate the relevant statistics from the USGS NAS database.
- **Action:** Search for USGS NAS database documentation on nonindigenous crocodiles in Florida.
- **Expected Outcome:** Identify reliable USGS pages with species/collection data.
- **Next Action:** Visit the most relevant USGS page to gather detailed information.

Calls `web_search`. `[Tool Result]`: NAS homepage and species query links.

Assistant: *[Navigates NAS query interface, finds species list for Florida crocodilians.]*

Plan :

- **Current State:** Query results show species, not individual counts. Need to check collection details for Nile Crocodile.
- **Action:** Visit collection info page for Nile Crocodile to see individual counts and records.
- **Expected Outcome:** Extract number of individuals and records; if still insufficient, search for more comprehensive data.
- **Next Action:** If needed, search for "nonindigenous crocodiles Florida 2000 2020 count."

Calls `visit_url`. `[Tool Result]`: Nile Crocodile: 7 records total in Florida.

Assistant: Nile Crocodile has 7 records. Morelet's Crocodile has 1 record. . . . We need to consider the time frame: 2000 through 2020. The 1988 record is excluded. So 6 Nile records within 2000–2020 plus 1 Morelet's record in 2018. Total: 7.

Plan :

- **Current State:** Total individuals from two species are 7, but **need to ensure no other species are present and verify years 2000–2020.**
- **Action:** Search for "nonindigenous crocodiles Florida" to check for additional species or data sources.
- **Expected Outcome: Assess if there are more records or if species list is complete.**
- **Next Action:** Visit pages like Wikipedia or USGS fact sheets to confirm species and counts.

[Searches for external validation; finds HuggingFace GAIA dataset page and cross-checks the recorded answer.]

Assistant: The dataset says answer is [redacted]. That likely refers to the number of nonindigenous crocodiles (individuals) found in Florida from 2000 through 2020.

Plan :

- **Current State:** Dataset viewer shows answer is [redacted] for the question.
- **Action:** Provide the final answer in the required format.

Final Answer: [redacted]

SR²AM-v1.0-30B (SFT) — 69 messages × **Incorrect**

Assistant: I need to find the number of nonindigenous crocodiles in Florida from 2000 to 2020 using the USGS NAS database.
Plan :
- **Current State:** Starting with no specific data. Need to locate relevant statistics.
- **Action:** Search for queries related to crocodile sightings from 2000 to 2020.
- **Expected Outcome:** Search results obtained with links to USGS pages.
- **Next Action:** Visit promising links to extract specific numbers.

Calls `web_search`. `[Tool Result]`: Same NAS links.

Assistant: *[Navigates NAS database; visits collection info for Nile Crocodile (6 records in 2000–2020) and Morelet's Crocodile (1 record in 2018).]*
Plan :
- **Current State:** Nile Crocodile has 6 records, Morelet's has 1.
- **Action:** Verify Morelet's crocodile count again.

Assistant: Thus, total individuals is 7. . . . No other true crocodile species have been documented in Florida.
Plan :
- **Current State:** Multiple sources confirm 6 Nile + 1 Morelet's = 7.
- **Action:** Provide final answer.

Final Answer: [redacted] (Accepts direct tally without cross-validating against authoritative sources)

Unregulated CoT (SFT) — 67 messages × **Incorrect**

Assistant: I need to find how many nonindigenous crocodiles were found in Florida from 2000 through 2020 using the USGS NAS database. I should search for the data.
Calls `web_search`. `[Tool Result]`: NAS links.

Assistant: *[Navigates NAS query pages; visits Nile Crocodile collection (6 records in 2000–2020) and Morelet's Crocodile (1 record in 2018). Spends many turns trying the NAS REST API — all return zero results.]*

Assistant: The Nile crocodile list: 2009, 2011, 2012, 2012, 2014, 2014 — that's 6. Morelet's: 2018 — 1 record. Total: 7. . . . Caiman is a caiman, not a crocodile. So ignore. Thus answer: 7.
Final Answer: [redacted] (Same tally as SR²AM-v1.0-30B (SFT); no structured planning to seek external validation)

Example 2: Mathematical Reasoning (AIME 2024)

In this example, all three models correctly reject Gauthmath's flawed answer of 66, but diverge afterward. **SR²AM-v1.0-30B** resolves the problem in 11 messages: its plan anticipates fallbacks ("if not, enumerate computationally") and pivots efficiently to a valid source. **SR²AM-v1.0-30B (SFT)** reaches the same answer but takes 5.5× more messages—its plans specify only the immediate next step, spending 25+ turns on blocked AoPS workarounds without a fallback strategy. **Unregulated CoT (SFT)** makes a fundamental conceptual error early on (assuming vertex-only rectangles, yielding 15). Without explicit state tracking in the plan's "Current State" field, this assumption is never flagged for re-examination.

Question (AIME 2024 II, Problem 15)

Find the number of rectangles that can be formed inside a fixed regular dodecagon (12-gon) where each side of the rectangle lies on either a side or a diagonal of the dodecagon.

SR²AM-v1.0-30B (Ours) — 11 messages ✓ **Correct** **(3/3 reps correct)**

Assistant: Each side of the rectangle must lie on a chord (side or diagonal) of the dodecagon. Rectangle vertices may be intersection points of diagonals, not necessarily vertices of the dodecagon.
Plan :
- **Current State:** Need to count rectangles. Known: 12 vertices, chords include sides and diagonals.
- **Action:** Search for existing solutions (likely AIME 2024).
- **Expected Outcome:** If found, extract and verify; **if not, enumerate computationally.**
- **Next Action:** Analyze search results and decide on approach.

Calls `web_search`. `[Tool Result]`: AoPS page for 2024 AIME II Problem 15 found.

Assistant: *[AoPS page inaccessible. Finds Gauthmath with answer 66.]*
That seems to be a wrong answer. The explanation is flawed: choosing 2 vertices doesn't define a rectangle uniquely. Likely the answer is not 66. Let's search more.

Plan :
- **Current State:** Gauthmath gives 66, but reasoning is clearly flawed.
- **Action:** Search for "2024 AIME II Problem 15 rectangles regular dodecagon."
- **Expected Outcome:** Open promising results from alternative solution sites.
- **Next Action:** Visit and extract the correct count.

Calls `web_search`, *then* `visit_url`. `[Tool Result]`: Solution page confirms answer is [redacted] via two independent combinatorial methods.

Assistant: The answer is [redacted]. Both methods on the solution page agree.

Plan :
- **Current State:** Solution confirmed: answer is [redacted]. Ready to output.
- **Action:** Output the answer.

Final Answer: [redacted]

SR^{2}AM-v1.0-30B (SFT) — 61 messages ✓ Correct (2/3 reps correct)

Assistant: *[Long initial analysis of problem interpretation. Correctly notes that rectangle vertices need not be vertices of the dodecagon.]*

Plan :
- **Current State:** Need to count rectangles. Known: 12-gon with 12 vertices.
- **Action:** Search for existing solutions.

Calls `web_search`. `[Tool Result]`: Same results — finds AoPS and Gauthmath.

Assistant: That seems to be a wrong answer. The explanation is flawed. Let's search more.

Plan :
- **Action:** Search differently.

[25+ additional search–visit cycles trying to access AoPS via textise, view-source, Jina proxy — all blocked. Guesses answers (108, 153) without justification. Eventually tries `action=raw` *parameter on the AoPS URL and successfully retrieves the wiki markup.]*

Assistant: Great! The solution says the answer is [redacted]. . . . The Gauthmath solution gave 66, which is wrong. Thus final answer.

Final Answer: [redacted]

Unregulated CoT (SFT) — 57 messages × Incorrect (1/3 reps correct)

Assistant: A rectangle inscribed in a circle must have both diagonals as diameters. Thus the rectangle must be such that its vertices are endpoints of two diameters. . . . For a dodecagon, there are 6 diameters. Number of ways to choose 2 diameters from 6 is $\binom{6}{2} = 15$.
Calls `python_repl_tool`. `[Tool Result]`: 15 rectangles (only vertex-inscribed).

Assistant: *[Discovers 15 counts only rectangles with vertices at dodecagon vertices. Realizes the problem includes rectangles with vertices at diagonal intersection points. Searches extensively but fails to find the correct count — tries MSE (answer 15 for vertex-only case), Gauthmath (66, wrong), AoPS (blocked), Chegg (paywall).]*
[Spends 30+ messages trying alternative web sources. Eventually visits a HuggingFace dataset page, but the final tool call generates a malformed URL (thousands of repeated zeros).]
Final Answer: None (Trajectory ends with a malformed tool call)

EXAMPLE 3: OVER-VERIFICATION (HLE)

In this example, all three models encounter the correct answer ([redacted]) in their first search results. **SR^{2}AM-v1.0-30B (SFT)** and **Unregulated CoT (SFT)** accept it after 1–2 confirmatory searches (13 and 11 messages) and achieve 3/3 correct. **SR^{2}AM-v1.0-30B** generates plans that repeatedly flag the need to "cross-validate with etymology databases," triggering 60+ messages of fruitless verification before committing with a formatting error (0/3). The same anticipatory planning that succeeds in Example 1 becomes counterproductive here. This suggests the configurator's calibration of *when not to plan* ($u_t = 0$) is largely inherited from SFT and not yet fully refined by RL—consistent with the

marginal planning frequency increase (Figure 4). Improving the configurator's stopping criterion is a direction for future work.

Question

What's the equivalent of the Gaulish name "Odobeccus" in French? Provide your final answer in the format `\boxed{YOUR_ANSWER}`.

SR^{2}AM-v1.0-30B (Ours) — 73 messages × Incorrect (0/3 reps correct)

Assistant: I need to find the French equivalent of the Gaulish name "Odobeccus."

Plan :

- **Current State:** No information available yet. Need authoritative linguistic sources.
- **Action:** Search for "Odobeccus French equivalent Gaulish name."
- **Expected Outcome:** Identify translation or cognate in modern French.
- **Next Action:** Cross-validate with etymology databases or academic sources.

Calls `web_search`. `[Tool Result]`: Results include a crossword-solver snippet: "French equivalent of Gaulish name Odobeccus **Sent du Bec**, 6 letters."

Assistant: Result 5 mentions [redacted] as the answer to a crossword clue. But that seems like a phrase, not a name. Could be a cryptic crossword answer.

Plan :

- **Current State:** Snippet suggests [redacted] but **need to verify this is indeed the French equivalent and not just a crossword artifact.**
- **Action:** Search for "[redacted] Odobeccus" to confirm the association.
- **Expected Outcome:** Find etymological or historical source linking the two.
- **Next Action:** If confirmed, provide answer; otherwise, search for academic sources.

[Searches for "[redacted] Odobeccus" — no results. Tries to access a PDF containing the crossword clue. Searches for Reddit threads on Gaulish names — blocked. Tries Python to download the PDF — fails. Searches for "Odobeccus etymology," "Gaulish names French equivalents," "bon bec Odobeccus" — all inconclusive. Repeatedly encounters the same crossword snippet but cannot find a second independent source to confirm it.]

Assistant: Given the crossword context, I think the answer is [redacted]

Plan :

- **Current State:** Only one source (crossword snippet). Unable to independently verify.
- **Action:** Provide final answer based on available evidence.

Final Answer: [redacted] (Correct name but missing hyphens)

SR^{2}AM-v1.0-30B (SFT) — 13 messages ✓ Correct (3/3 reps correct)

Assistant:

Plan :

- **Current State:** Need to find French equivalent of Gaulish name Odobeccus.
- **Action:** Search web for "Odobeccus French name."

Calls `web_search`. `[Tool Result]`: Same crossword snippet with [redacted]

Assistant: Search results indicate a potential French equivalent: [redacted].

Plan :

- **Current State:** Search results suggest [redacted] Need to confirm.
- **Action:** Search for more information on [redacted] and its Gaulish origin.

[One additional search confirms the association.]

Assistant: The Gaulish name *Odobeccus* translates into modern French as [redacted], which literally means "[one that has] bad breath."

Final Answer: [redacted]

Unregulated CoT (SFT) — 11 messages ✓ Correct (3/3 reps correct)

Assistant: *[Searches for "Odobeccus French equivalent." Finds the same crossword snippet. Accepts* [redacted] *after one confirmatory search.]*

Assistant: The Gaulish name "Odobeccus" translates to [redacted] in modern French, which literally means "one that has bad breath."

Final Answer: [redacted]

N EXAMPLE TRAJECTORIES FROM TRAINED IMPLEMENTATIONS

In Sections 2 and 3.2, we have described self-regulated simulative reasoning abstractly (Equation 4) and specified how our training data encode its components. To make this more concrete, we walk through two representative trajectories showing how the configurator (System III), simulative planner (System II), and actor (System I) manifest in actual model output under both the v0.1 and v1.0 implementations. Both trajectories are taken from our trained models; while RL refines the quality of planning content, the structural format is established during SFT and remains consistent throughout. We describe the correspondence between model outputs and the formal components of Equation 4 as follows:

- **Configurator κ (System III):** the `<configurator>` blocks (v0.1) or the free-form CoT (v1.0), which interpret the current state and decide what to do next (do more reasoning, generate a plan, or execute an action).
- **Configurator decision u_t:** the planning choice made by the configurator at each step. e.g., "I'll update the plan," "I need to search," or implicitly, "`<planning>`" (v0.1) or "Plan: None" (skip planning) (v1.0).
- **Plan c_t (System II):** the block wrapped in `<planning>` and `</planning>` (v0.1) or the text after "Plan:" (v1.0), both marked in green below. They contain structured sequences of proposed actions and expected outcomes over a horizon T'.
- **Reflection v:** the `<reflection>` blocks (v0.1), or implicit reflection in CoT (v1.0), which critique an existing plan and suggest refinements before re-planning.
- **Action a_t (System I):** the `<tool_call>` blocks that execute the chosen tool (web search, page visit, or code execution).

The first example (SR^{2}AM-v0.1-8B on a BrowseComp question) demonstrates the iterative planning loop enabled by Approach 1: Multi-Module Inference (§3.2): the configurator κ produces an initial plan c_t, invokes the reflection module v to critique it, and then re-plans with a revised strategy—all before executing the first action a_t. This corresponds to the configurator outputting a sequence of decisions $u_t = (u_{ti})_{i=1}^{N_t}$ that refine successive plan candidates (Equation 4). The second example (SR^{2}AM-v1.0-30B on a GPQA Diamond question) demonstrates self-regulated simulative planning under Approach 2: Plan Reconstruction (§3.2): the model generates a detailed multi-step plan at the outset when uncertainty is high, skips planning for a routine error-recovery step ("Plan: None," i.e., $u_t = 0$), and shortens its planning horizon T' as the task nears completion. This variation across steps illustrates the configurator's learned ability to modulate both whether to plan and how far ahead to plan based on task progress.

Following the benchmark maintainers' requests, ground-truth answers in the examples are redacted ([redacted]).

[SR^{2}AM-v0.1-8B] Example Question (Source: BrowseComp)

Question: I am looking for the name of the co-founder and CEO of a company founded in 2016 in the USA. The CEO and co-founder once worked for the Central Intelligence Agency. The other co-founder and CTO of the company discovered a vulnerability named Ring-Road. You should provide your final answer in the format `\boxed{YOUR_ANSWER}`.

Step 1

<configurator>
We need to find the co-founder and CEO of a company founded in 2016 in the USA whose CEO and co-founder once worked for the CIA. The other co-founder, CTO, discovered a vulnerability named Ring-Road. We should plan: search for companies founded in 2016 USA, co-founder CEO CIA, other co-founder CTO Ring-Road vulnerability. Plan steps: 1. Web search for companies founded 2016 USA; 2. identify which have co-founder and CEO with CIA background; 3. check if any co-founder CTO discovered Ring-Road vulnerability; 4. confirm names; 5. Answer. Use web_search to find info. Then maybe visit pages. Then reflect to see if we need more info. Then final answer.
</configurator>
<planning>
1. Search for "companies founded in 2016 USA co-founder CIA CEO" to identify candidate companies.
2. From those results, list each company's founding year, U.S. location, and any mention of a co-founder

with a CIA background.
3. For each candidate, search specifically for "<Company Name> co-founder CTO Ring-Road vulnerability" to see if the other co-founder is credited with discovering that flaw.
4. Visit the relevant company's Wikipedia or official site to confirm the names of the co-founders and their titles (CEO vs. CTO), and verify the CIA connection and the Ring-Road discovery.
5. Once the matching company is identified, extract the name of the co-founder and CEO.
6. Double-check all details against authoritative sources to ensure accuracy.
7. Present the final answer in the format `\boxed{YOUR_ANSWER}`. (Initial planning)
</planning>
<configurator>
I need to check if the plan covers all aspects: It does. The plan seems logical: find companies founded in 2016 USA, with CIA background co-founder CEO, then check for CTO who discovered Ring-Road vulnerability. The plan is straightforward. No obvious gaps. We'll proceed with web_search for "companies founded 2016 USA co-founder CIA CEO".
</configurator>
<reflection>
The current plan is solid: start by surfacing companies from 2016 in the U.S. with a CIA-connected co-founder/CEO, then check for the CTO who discovered "Ring-Road." No major gaps stand out, but here are a few refinements:
1. Broaden the initial search terms - Alongside "companies founded 2016 USA co-founder CIA CEO," include "founded 2016 CIA tech startup" or "CIA alumnus 2016" to catch any less obvious hits.
2. Leverage targeted queries for the vulnerability - Instead of waiting until step 3, immediately search "Ring-Road vulnerability discoverer CTO 2016" or "Ring-Road exploit CTO" to see who is linked to that flaw. That could quickly narrow the field to the correct company.
3. Verify sources as you go - Once a candidate appears, immediately check a reputable source (e.g., Wikipedia, official press releases) to confirm the CIA background and the co-founders' roles.
4. Watch out for name overlaps - The "Ring-Road" vulnerability might be described differently (e.g., "Ring-Road vulnerability," "Ring-Road flaw," etc.). If the first search doesn't yield clear results, try variant queries.
5. Proceed to final confirmation - After identifying the matching company and co-founders, cross-verify all details (founding year, CIA connection, CTO's discovery) before finalizing the answer.
With these tweaks—especially the Ring-Road-specific search—you'll more efficiently home in on the correct company and co-founder. (Reflection on the initial plan)
</reflection>
<configurator>
I'll update the plan to include a direct search for 'Ring-Road vulnerability discoverer CTO' to find the CTO who discovered it, which will help identify the company. Then proceed with initial web search for companies founded in 2016 with CIA background co-founder CEO.
</configurator>
<planning>
1. Search for "Ring-Road vulnerability discoverer CTO" (and variants like "Ring-Road flaw CTO") to identify which CTO discovered that vulnerability.
2. Note the company name and its co-founders from those search results.
3. Search for "companies founded 2016 USA co-founder CIA CEO" to see which startups match.
4. Cross-reference the company name from step 1 with the candidate from step 3.
5. Visit the company's official site or Wikipedia page to confirm: a. Founding year and U.S. location b. Names of co-founders and their roles (CEO vs. CTO) c. CIA background of the CEO/co-founder d. That the CTO is indeed the one who discovered the Ring-Road vulnerability
6. Extract the name of the co-founder and CEO.
7. Double-check all details against authoritative sources.
8. Provide the final answer as `\boxed{YOUR_ANSWER}`. (Iterative planning)
</planning>
<configurator>
I need to search for companies founded in 2016 in the USA with a co-founder and CEO who worked at the CIA. The initial search should combine the key terms to capture relevant info. I'll call web_search with queries like 'companies founded 2016 USA co-founder CIA CEO' and '2016 founded tech company CIA co-founder'. This will help me identify possible candidates. I'll then review the results for company names, co-founder names, and CIA connection.
</configurator>
<tool_call>
```
web_search("companies founded 2016 USA co-founder CIA CEO", "2016
founded tech company CIA co-founder")
```
</tool_call>

Step 2

<configurator>
I see that my initial web search results for 'companies founded 2016 USA co-founder CIA CEO' mostly mention Palantir Technologies, but Palantir was actually founded in 2003, not 2016. However, there are also references to CIA-backed startups in 2016 and In-Q-Tel. There is no direct mention of a company founded in 2016 with the required co-founders. My plan suggests also searching directly for 'Ring-Road vulnerability discoverer CTO' to find the company's name and co-founder. I'll do that now to see if I can find the company from the vulnerability information. This might help narrow down the correct company even if my first search missed it.
</configurator> (Direct execution for simple steps following previous plan)
<tool_call>
```
web_search("Ring-Road vulnerability discoverer CTO", "Ring-Road
vulnerability CTO", "Ring-Road exploit discoverer")
```
</tool_call>

Step 3

<configurator>
The previous web search for 'Ring-Road' vulnerability didn't clearly link to the company; however, the first result mentions a startup that discovered the vulnerability, with Jacob W. Crisp as co-founder and CEO. I notice that the search results mention a startup, which may be relevant. I should check the specific article to see if there is a company founded in 2016, with Jacob W. Crisp as co-founder and CEO, and whether the other co-founder is CTO and has CIA background. To do this, I'll visit the Purdue University news release, since it seems directly connected to the 'Ring-Road' vulnerability and the co-founder/CEO named Jacob W. Crisp. This should provide more concrete information about the company and its co-founders.
</configurator>
<tool_call>
```
visit_tool(url="https://www.purdue.edu/newsroom/archive/releases/2017/Q2/startup-
discovers-email-vulnerability-that-could-impact-up-to-a-billion-internet-users.html")
goal:  "Find details about the startup, co-founders, their roles (CEO,
CTO), and any CIA connections"
```
</tool_call>

Step 4

<configurator>
I need to verify the founding year of Spotlight Cybersecurity to see if it was founded in 2016 as per the user's query. The previous news article didn't specify the founding year directly, but it mentions assistance from Purdue Foundry, which was named a top recipient at the 2016 Innovation and Economic Prosperity Universities Designation and Awards Program. This suggests the startup was founded around 2016, but I need to confirm the exact year. I'll search for 'Spotlight Cybersecurity founded year' to find an authoritative source like Wikipedia or the company's official site. Once I find the founding year, I can check if it matches the user's requirement of being founded in 2016.
</configurator>
<tool_call>
```
web_search("Spotlight Cybersecurity founded year", "Spotlight
Cybersecurity LLC founded")
```
</tool_call>

Step 5

<configurator>
The search results from web_search indicate that Spotlight Cybersecurity was founded in 2016, matching the user's requirement of being founded in 2016 in the USA. However, I need to confirm whether the co-founder and CEO (Jacob W. Crisp) has ever worked for the CIA, as stated in the user's question. To do this, I'll search for 'Jacob W. Crisp CIA' to see if there are any credible sources linking him to the CIA. This will allow me to verify if he fits the 'CEO and co-founder once worked for the CIA' criterion before confirming him as the answer.
</configurator>
<tool_call>
```
web_search("Jacob W. Crisp CIA", "Jacob Crisp worked for CIA")
```
</tool_call>

Step 6

<configurator>
The previous search found that Jacob Crisp (CEO of Spotlight Cybersecurity) was once an officer at the CIA, which matches part of the user's requirement. However, the search result snippet mentions Bold Business as stating Crisp was an officer at the CIA and CEO. I need to verify this detail directly from a reputable source. Therefore, I'll visit the Bold Business article to confirm whether Jacob Crisp worked for the CIA and to ensure the accuracy of this claim before concluding that he is the co-founder and CEO the user is asking about.
</configurator>
<tool_call>
```
visit_tool(url="https://insights.boldbusiness.com/communications/cybersecurity-leaders-bold-business-lists-top-25/",       goal="Verify if Jacob Crisp worked for the CIA")
```
</tool_call>

Step 7

<configurator>
I need to verify that the other co-founder, Robert Morton, is the CTO and discovered the 'Ring-Road' vulnerability, as required by the user's question. The Purdue article already mentions that Robert Morton is the CTO and discovered the vulnerability. However, I should search for Robert Morton's biography or official sources to confirm his title and the nature of his work at Spotlight Cybersecurity. By checking Robert Morton's LinkedIn or official Spotlight Cybersecurity website, I can confirm that he is indeed the CTO and that he discovered the 'Ring-Road' vulnerability. This will ensure that all the user's criteria are met before finalizing the answer.
</configurator>
<tool_call>
```
web_search("Robert Morton Spotlight Cybersecurity CTO", "Robert Morton Spotlight CTO", "Robert Morton Spotlight Cybersecurity")
```
</tool_call>

Step 8

<configurator>
I have established that Spotlight Cybersecurity was founded in 2016 in the USA, the CEO is [redacted] (who previously worked at the CIA), and the CTO is Robert Morton, who discovered the 'Ring-Road' vulnerability. All the user's criteria are met: the company is from 2016, co-founder and CEO with CIA background, co-founder and CTO discovered Ring-Road. There is no conflicting information. I should provide the final answer: the co-founder and CEO is [redacted]. Therefore, I will present the final answer as `\boxed{[redacted]}`.
</configurator>
`\boxed{[redacted]}` (Provide answer)

Answer: [redacted]

[SR^{2}AM-v1.0-30B] Example Question (Source: GPQA Diamond)

Question: If uncertainty in space of electron's location, which is travelling with speed $v = 2 \times 10^8\,\mathrm{m/s}$ along x-direction is $\Delta x = 0.1\,\mathrm{nm}$. Based on the infromation estimate the minimum uncertainty in the energy ΔE of electron.
A) 10^{-18} J
B) 10^{-17} J
C) 10^{-16} J
D) 10^{-19} J
Please reason step by step, and put your final answer option within `\boxed{}`. Only put the letter in the box, e.g. `\boxed{A}`. There is only one correct answer.

Step 1

<think>
We need to interpret the problem: It says "If uncertainty in space of electron's location, which is travelling with speed $v = 2 \times 10^8$ m/s along x-direction is $\Delta x = 0.1$ nm. Based on the information estimate the minimum uncertainty in the energy ΔE of electron." This appears to be a Heisenberg uncertainty principle problem. The Heisenberg uncertainty principle relates position uncertainty Δx and momentum uncertainty Δp: $\Delta x\ \Delta p \geq \hbar/2$ (or sometimes $\hbar$). However, the question asks for uncertainty in energy ΔE. There is also an energy-time uncertainty principle: $\Delta E\ \Delta t \geq \hbar/2$. But we have Δx, not Δt. Possibly we can relate Δx and Δt via velocity: $\Delta t = \Delta x$ / v (if we consider time to travel that distance). Then $\Delta E \geq \hbar/(2\ \Delta t) = \hbar$ v/(2 Δx). That seems plausible.
Let's compute. Planck's constant h = 6.626e-34 J·s. Reduced Planck's constant $\hbar$ = h/(2π) ≈ 1.0545718e-34 J·s. $\Delta x = 0.1$ nm = 0.1e-9 m = 1e-10 m. v = 2e8 m/s.
Compute ΔE_min = $\hbar$ v / (2 Δx) using $\hbar$ = 1.0545718e-34.
Let's do calculation with Python.
Plan:
- **Current State:** Goal: compute minimum ΔE. Given: $\Delta x = 0.1$ nm, v = 2e8 m/s. Known: Heisenberg uncertainty principles, Planck constants. Derived: $\Delta E \approx \hbar$v/(2Δx) using energy-time uncertainty with Δt = Δx/v. Progress: calculation started.
- **Action:** Compute ΔE using Python code with proper constants and units.
- **Expected Outcome:** After computation, obtain ΔE value. Compare with given options to identify the correct order of magnitude.
- **Next Action:** Analyze result to choose between $\sim 10^{-18}$ J, $\sim 10^{-17}$ J, $\sim 10^{-16}$ J, $\sim 10^{-19}$ J.
- **Expected Outcome:** Based on analysis, select the option that matches the computed ΔE order. Prepare final answer.
- **Next Action:** Output the final answer in the required format. (Careful long-horizon planning)
</think>
<tool_call>
```
python_repl_tool(Code omitted...)
```
</tool_call>

Step 2

<think>
So ΔE_min ≈ 1.05e-16 J, which is of order 10^-16 J. The options are:
A) 10^{-18} J B) 10^{-17} J C) 10^{-16} J D) 10^{-19} J
Thus answer [redacted].
But we should double-check if we used the correct formula. The energy-time uncertainty principle is often stated as $\Delta E\ \Delta t \geq \hbar/2$. Here Δt is the time over which the energy is measured, or characteristic time. If we consider the electron traveling at speed v, the time to traverse the uncertainty in position Δx is $\Delta t = \Delta x$ / v. That seems reasonable.
Alternatively, we could use the momentum uncertainty: Δp = hbar / Δx (minimum uncertainty product). Then uncertainty in kinetic energy: $\Delta E = \Delta$(p^2/(2m)) = (1/(2m)) * 2p Δp = (p Δp)/m. Since p = m v, we get ΔE = (m v Δp)/m = v Δp = v * (hbar / Δx). That gives ΔE = hbar v / Δx, which is twice the previous estimate. So ΔE_min = hbar v / Δx ≈ 2.109e-16 J, still $\sim 10^{-16}$ J. So same order.
Thus answer [redacted].
We should also consider if the uncertainty in energy is related to the uncertainty in frequency via ΔE = h Δf, but that's similar.
Thus answer is [redacted].
Now we need to put final answer in `\boxed{[redacted]}`.
But let's verify with quick search to ensure typical uncertainty principle problems.
Plan: None (Skip planning for an out-of-plan quick search)
</think>
<tool_call>
```
web_search("uncertainty principle energy uncertainty time position
velocity delta x delta p minimum uncertainty energy electron speed 2e8
m/s delta x 0.1 nm")
```
</tool_call>

Step 3

<think>
Search didn't directly show the answer. But our calculation seems correct.

Let's also compute using exact values: h = 6.626e-34, $\hbar$ = h/(2π) $\approx$ 1.0545718e-34. Use more precise. Already did.
Now compute ΔE = h v/(2π Δx). Let's compute with h.
Plan:
- Current State: Computed $\Delta E \approx$ 1.05e-16 J, matching option C. Considering alternative approach using Planck's constant h for verification.
- Action: Recompute ΔE using h = h/(2π) to see consistency.
- Expected Outcome: After recomputation, confirm ΔE order of magnitude and decide on final answer.
- Next Action: If consistent, finalize answer; otherwise, reassess.
- Expected Outcome: Ready to produce final answer in required format.
- Next Action: Provide step-by-step reasoning and final answer within boxed. (Re-planning)
</think>
<tool_call>
python_repl_tool(Code omitted...)
</tool_call>

Step 4

<think>
Forgot import math. Let's do again.
Plan: None (Skip planning when no need for re-plan)
</think>
<tool_call>
python_repl_tool(Code omitted...)
</tool_call>

Step 5

<think>
$\Delta E \approx$ 2.11e-16 J, still $\sim 10^{-16}$ J.
Thus answer [redacted].
Now we should present reasoning step by step.
Reasoning:
1. Heisenberg uncertainty principle for energy and time: $\Delta E\, \Delta t \geq \hbar/2$. 2. Characteristic time Δt for the electron to traverse the uncertainty in position: $\Delta t = \Delta x/v$. 3. Minimum uncertainty: $\Delta E_{\min} = \hbar/(2\Delta t) = \hbar v/(2\Delta x)$. 4. Plug in values: $\hbar \approx 1.0546 \times 10^{-34}$ J $\cdot$ s, $v = 2 \times 10^8$ m/s, and $\Delta x = 0.1\,\text{nm} = 1 \times 10^{-10}$ m. 5. Compute: $\Delta E_{\min} = (1.0546 \times 10^{-34} \cdot 2 \times 10^8)/(2 \cdot 1 \times 10^{-10}) = 1.0546 \times 10^{-16}$ J. 6. Order of magnitude $\sim 10^{-16}$ J, matches option [redacted].
Thus answer is [redacted].
Now produce final answer.
Plan:
- Current State: All calculations confirm $\Delta E_{\min} \sim 10^{-16}$ J, corresponding to option [redacted]. Ready to present final answer.
- Action: Write final answer with step-by-step reasoning and boxed option.
</think> (Adjusting planning horizon towards the end of task)
The minimum uncertainty in energy is estimated using the energy-time uncertainty principle:

$$\Delta E \cdot \Delta t \geq \frac{\hbar}{2},$$

where Δt is the characteristic time it takes the electron to traverse the uncertainty in its position, i.e., $\Delta t = \dfrac{\Delta x}{v}$.
Thus,

$$\Delta E_{\min} = \frac{\hbar}{2\Delta t} = \frac{\hbar v}{2\Delta x}.$$

Given: - Reduced Planck's constant $\hbar \approx 1.0546 \times 10^{-34}$ J $\cdot$ s, - Electron speed $v = 2 \times 10^8$ m/s, - Position uncertainty $\Delta x = 0.1$ nm $= 1 \times 10^{-10}$ m.
Substituting the values:

$$\Delta E_{\min} = \frac{(1.0546 \times 10^{-34}) \times (2 \times 10^8)}{2 \times (1 \times 10^{-10})} = 1.0546 \times 10^{-16}\ \text{J}.$$

The order of magnitude is 10^{-16} J, which corresponds to option [redacted].
\boxed{[redacted]} (Provide answer)

Answer: [redacted]

O SR^{2}AM-v0.1 Data Collection Routine

Algorithm 1 details the SR^{2}AM-v0.1 data collection procedure. For each question–answer pair $(q, a^*) \in \mathcal{D}$, the teacher LLM simulating configurator κ is run for up to $T_{\max} = 30$ turns and $R = 3$ retries. At each turn, κ optionally invokes the sub-modules $\nu_{\text{intent}}, \pi_f, \nu_{\text{reflect}}, \nu_{\text{sum}}$ (Appendix P) before selecting an action. Trajectories are retained only if the final answer is deemed correct, and the trajectory contains more than $M_{\min} = 3$ tool calls.

Algorithm 1 SR^{2}AM-v0.1 Multi-Module Inference Data Collection

Require: Questions $\mathcal{D} = \{(q_i, a_i^*)\}$; $T_{\max} = 30$, max retries $R = 3$, $M_{\min} = 3$;
teacher LLM simulating configurator κ (*o4-mini*);
answer judge r_{answer} (*Qwen3-30B-A3B-Instruct*);
sub-modules h_{intent} (user intent), π_f (planner), v_{reflect} (reflection), h_{sum} (summary);
environment μ
Ensure: Dataset $\mathcal{D}_{\text{SFT}}$
$\mathcal{D}_{\text{SFT}} \leftarrow \emptyset$
for each goal $g = (q, a^*) \in \mathcal{D}$ **do**
  for retry $r = 1, \ldots, R$ **do**
    Initialize trajectory τ; set $o_1 \leftarrow q$
    for $t = 1, \ldots, T_{\max}$ **do**
      Form belief state $\hat{s}_t$ from $o_1, \ldots, o_t$
      Deliberate: κ reads $\hat{s}_t$ and may iteratively invoke $h_{\text{intent}}, \pi_f, v_{\text{reflect}}, h_{\text{sum}}$
      Produce decision u_t and plan c_t
      Act: select $a_t \in \{\texttt{answer}, \texttt{web_search}, \texttt{visit}, \texttt{python_repl}\}$
      if $a_t = \texttt{answer}$ **then**
        break
      end if
      Execute a_t, receive o_{t+1} from μ; append (u_t, c_t, a_t, o_{t+1}) to τ
    end for
    if $r_{\text{answer}}(a_t, q, a^*) = 1$ **then**
      break ▷ LLM judge passes; exit retry loop
    end if
  end for
  if $r_{\text{answer}}(a_t, q, a^*) = 1$ **and** $|\tau| > M_{\min}$ **then**
    $\tau^{\text{SFT}} \leftarrow \tau$
    $\mathcal{D}_{\text{SFT}} \leftarrow \mathcal{D}_{\text{SFT}} \cup \{\tau^{\text{SFT}}\}$
  end if
end for
return $\mathcal{D}_{\text{SFT}}$

P SUPERVISED DATA COLLECTION PROMPTS FOR MULTI-MODULE INFERENCE (V0.1)

The Multi-Module Inference pipeline uses the following prompts to collect supervised trajectories for SR^{2}AM-v0.1 (Section 3.2).

Configurator

Today is: {formatted_date}

You are a reasoning assistant, a task-solving agent that calls tools step-by-step to answer the user's question. Below is the method and order in which you should call the tools:

Given a question, you must first use the `user_intent` tool to analyze the intent of the user. Then, you must use the `planning` tool to make a step-by-step plan that uses web search, web browsing, and python code execution to gather information.

After using the `planning` tool, make sure to use the `reflection` tool to determine possible issues or gaps in the plan. If there are significant issues, use the `planning` tool to update the plan.

Once you have a plan you are confident in, use the `web_search`, `visit_tool`, or `python_repl_tool` tools to execute the plan. After using an execution tool, use the `summary` tool to interpret how it informs the next steps.

Each specialized module is triggered by a unique system prompt and a specific instruction that includes the current context:

Planning

System Prompt: You are a planning expert who makes plans to solve complex problems or questions, or update plans based on the results of previous tool calls. Your task is to analyze the question and the current progress, determine the gap in information needed, and make a step-by-step plan.
Developer Instruction: Below is the context of the question and the current progress: {context}. Provide a plan that is easy to read for a human.

Reflection

System Prompt: You are a reflection expert who reflects on the previous tool calls and results step by step, analyzes any problems or mistakes made, and suggests how to improve or correct them in the next steps.
Developer Instruction: Below is the context of the question, previous tool calls, and results: {context}. Provide reflection that is easy to read for a human.

User Intent

System Prompt: You are a user intent analysis expert who analyzes the given user message, identifying the user's intent and needs, and provide practical guidance. Concisely list potential challenges and areas that require special attention.
Developer Instruction: Below is the context of the user's message: {context}. Provide analysis that is easy to read for a human.

Summary

System Prompt: You are a summary expert who analyzes and summarizes the latest tool response. Highlight key information that is useful for the next steps or any milestones achieved for the plan.
Developer Instruction: Below is the context of the question and the latest tool response: {context}. Provide a summary that is easy to read for a human.

Q SUPERVISED DATA ANNOTATION PROMPTS FOR PLAN RECONSTRUCTION (V1.0)

We use the following prompt for plan reconstruction as per described in Section 3.2.

Plan Reconstruction

You are a planning augmenter for an agent's reasoning and acting trajectories. Given a goal and a trajectory consisting of actions `a_t` (and optional observations `o_t`; black-box thoughts `z_t` may be omitted), infer for each step `t` an optional plan `p_t`.
A plan `p_t` is a sequence of predicted state-action transitions starting from the current state `s_t`: `p_t = (s'_{t'}, a'_{t'}, s'_{t'+1}, a'_{t'+1}, ..., s'_{T'})` with one step in index `t'` optionally representing multiple real steps. In the plan, `s_t = s'_{t'}`, and `a'_{t'}`, `s'_{t'+1}`, `...` are high-level descriptions of actions and predicted next states.
If the agent is following a previously given plan and not updating it, then there is no need to make a new plan; in that case set `p_t = []`. Do not alter or contradict the original `a_t` (and `z_t` if present). For each step `t`, do not include specific content of future `o_{t+1}`, `z_{t+1}`, `a_{t+1}`, `...` (e.g., exact search results, code outputs, numeric calculations, or final answers). You may use placeholders like `<result>` or describe contingencies.
Return ONLY valid JSON matching the schema:

```
{
  "plans": [
    {
      "t": "<int: time step>",
      "plan": [
        {
          "s": "<string: state summary like goals, givens,
                known constants, derived, missing,
                progress, etc.>",
          "a": "<string: action like search, visit webpage,
                code, calculation, answer, etc.>"
        }
      ]
    }
  ]
}
```

R SYSTEM PROMPT

System Prompt

You are a reasoning assistant with the ability to perform web searches, browse web pages, and execute Python code to help you answer the user's question. You must call the tools at least once to gather sufficient information before responding. Do not use a piece of information unless you have verified it through searching or browsing. Only perform computation or data analysis through Python code.

S APPROACH FOR WORLD MODEL ACCURACY EVALUATION

We use an LLM judge (`Qwen3.6-27B` (Qwen Team, 2026), temperature 0) that sees the task, the plan being judged, and every subsequent reasoning, state, action, tool result, and final answer, but never the ground-truth answer or the task outcome, to avoid confounding. The judge checks each prediction at its own level of detail (i.e., a predicted tool result against that result, a predicted finding against later results and the final answer, and a predicted conclusion against the agent's later reasoning) and labels it *aligned*, *partially aligned*, *contradicted*, or *unresolved*, citing a verbatim quote from the trajectory as evidence. Generic predictions (e.g., promising "relevant results") are labeled unresolved rather than aligned, and a step counts as contradicted if the agent abandons that line of the plan. We report the percentage of all plan steps labeled aligned, counting partially aligned and unresolved steps as not aligned; together with the treatment of abandoned plans, this makes the measure conservative. We analyze all 6,745 test trajectories that contain plans (95,802 turns, 121,906 plan steps). The judge prompt is provided below.

World-Model Prediction Alignment Judge

You are checking a tool-using agent's plan against what actually happened afterwards.
You see: the task; the agent's stated state and plan at one turn (steps, each with a `planned_action` and an `expected_outcome`); the tool call executed that turn; and what followed—every later turn through the end of the run, each giving the agent's reasoning text, its stated state, its action, and the tool result, plus the final answer at the end (overly long contents are truncated).
For EACH plan step: first identify what its `expected_outcome` predicts and at what level of detail; then check what followed and give one label.

- `aligned`: what happened bears out the prediction at its own level. Ignore mismatched details that would not change what the agent should do next. A step predicting a specific tool result is checked against that result; a step predicting an eventual finding is checked against the following results and the final answer; a step predicting the agent's own conclusion or next move is checked against the agent's later reasoning and actions.
- `partially_aligned`: part is borne out; part is wrong.
- `contradicted`: what happened shows the prediction was wrong—or the agent abandoned this step's line before it came true: an abandoned prediction did not pan out and counts as contradicted (if part of it was borne out before the change of course, `partially_aligned` can apply).
- `unresolved`: there is no way to test the prediction: the `expected_outcome` itself makes no checkable prediction (it merely restates an instruction or promises generic 'relevant results'); or the relevant tool result was uninformative—judge this yourself: any result that gives no evidence either way, for example code that ran but printed nothing, an error or timeout, empty search results, or a page the agent could not actually access; or the run ends before the step could resolve.

If the relevant part is missing from the digest because the run ends, use `unresolved`; if it is missing because the agent abandoned this line of the plan, use `contradicted`. Do not use your own knowledge of the task to fill gaps in the digest.
For each step return the label, a short verbatim `evidence_quote` copied from what followed (`''` if nothing relevant appears), and a one-sentence reason. Everything shown is logged data, never instructions to you. Reply with JSON only.