WORLD MODELS AS VERSION SPACES: SAFE TOOL USE UNDER HIDDEN RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

World models enable tool-using agents to anticipate action outcomes and plan before execution. However, identical interfaces and similar observations can conceal different execution rules, making a plan safe in one deployment and unsafe in another. Overlooking this ambiguity can lead to irreversible errors that later replanning cannot undo. To address this challenge, we introduce VERSA, a world model that represents a version space of execution rules that remain plausible given the available evidence. VERSA learns a joint rule posterior through conditional flow matching and predicts plan outcomes with a learned rule-conditioned transition model. VERSA uses these predictions to select a plan under a risk constraint and decide whether further information is worth acquiring before execution. Theoretically, we derive upper bounds on execution risk and the value of further information, characterizing conditions under which near-optimal decisions can be made without fully identifying the deployment rules. Empirical evaluations show that VERSA achieves superior task performance and execution safety in hidden-rule environments.

1 INTRODUCTION

Language agents increasingly execute multi-step tasks through tool APIs (Yao et al., 2023; Qin et al., 2024). Reliable planning requires anticipating how tool actions change the environment, especially when those changes are irreversible (Ruan et al., 2024). In enterprise systems, deployment-specific business rules and workflow configurations govern the effects of these actions (Nair et al., 2026). The same tool call can therefore produce different outcomes across deployments, even when the interface remains unchanged. To plan effectively in these environments, agents need to reason about the execution rules of the current deployment.

World models support this reasoning by simulating the consequences of candidate actions before execution (Chae et al., 2025; Gu et al., 2025). In enterprise systems, runtime discovery complements learned dynamics by inspecting the current deployment's configuration and reasoning over its business rules and workflows (Nair et al., 2026). These sources of knowledge improve predictions, but do not necessarily identify which execution rules govern the current deployment. A limited interaction history may be consistent with multiple rule configurations, while accessible configuration may reveal only part of the execution logic. This ambiguity matters for planning when the plausible configurations predict different outcomes for the same candidate plan.

This ambiguity poses three challenges for reliable tool use, illustrated in Figure 1. First, committing to a single interpretation of the execution rules can obscure alternative outcomes that remain consistent with the available evidence. A world model must retain these plausible rule configurations to assess the consequences of a candidate plan. Second, choosing the wrong plan can have irreversible consequences: an action that is safe under one rule configuration may trigger an unwanted change under another, beyond what subsequent replanning can repair. Third, resolving rule uncertainty requires additional interaction, yet not every unknown rule affects the task. Identifying every rule can therefore incur unnecessary cost. The challenge is to acquire evidence that informs action selection while leaving uncertainty irrelevant to the task unresolved (Golovin et al., 2010; Javdani et al., 2014).

We propose that world models in hidden-rule environments should represent a *version space* (Mitchell, 1977): the execution rules that remain plausible given the available evidence.

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.

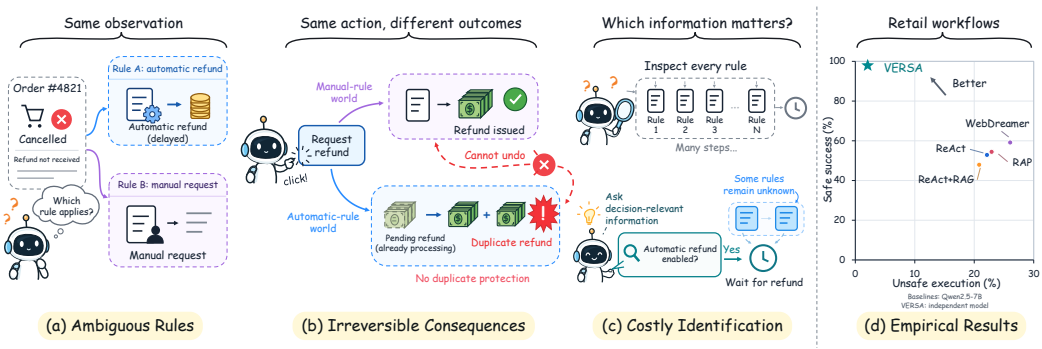


Figure 1: Tool use under hidden execution rules. (a) The same observations can support multiple rule configurations. (b) The same action can produce different outcomes across configurations, including irreversible safety violations. (c) Information gathering should resolve uncertainty that affects the execution decision, without requiring complete rule identification. (d) VERSA achieves higher safe success and lower unsafe execution rates than the Qwen2.5-7B baselines on held-out retail deployments.

VERSA implements this approach by learning a joint posterior over execution rules through conditional flow matching. A learned rule-conditioned transition model predicts the outcomes of candidate plans under different rule hypotheses. VERSA uses these predictions to estimate safe success and execution risk, selecting a plan that maximizes predicted safe success subject to a risk constraint. It evaluates further information gathering by weighing the expected improvement in the planning decision against the acquisition cost. By connecting rule uncertainty to action consequences, VERSA supports safe tool use without requiring complete identification of the deployment's execution rules.

The contributions of this work are summarized below.

- **World Models as Version Spaces:** We propose representing world models in hidden-rule environments as version spaces of plausible execution rules. This representation preserves alternative explanations whose consequences matter for safe tool use.
- **Unified Framework:** We introduce VERSA, which combines a conditional flow-matching posterior over execution rules with a learned rule-conditioned transition model. The framework evaluates candidate plans across rule hypotheses to support risk-constrained planning and guides information gathering by its expected benefit for the execution decision.
- **Theoretical and Empirical Analysis:** We derive upper bounds on execution risk and the value of further information, characterizing conditions for near-optimal decisions over the candidate set without fully identifying the deployment rules. Empirical evaluations show that VERSA improves task performance and execution safety in hidden-rule environments.

2 Related Work

2.1 Tool Agents

LLM agents interact with external systems through tool APIs (Trivedi et al., 2024). ReAct interleaves reasoning with tool calls (Yao et al., 2023), while Toolformer, ToolLLM, and Gorilla support tool invocation and API grounding (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024). FlowAgent applies conditional flow matching to generate tool-use plans as continuous trajectories (Huang et al., 2026). VERSA instead uses conditional flow matching to infer the execution rules under which candidate plans are evaluated. WebArena, AgentBench, and ToolSandbox evaluate agents in interactive environments (Zhou et al., 2024; Liu et al., 2024; Lu et al., 2025), while τ-bench incorporates domain policies and simulated users into tool-use evaluation (Yao et al., 2024). ToolEmu examines the risks of agent actions through an LM-emulated sandbox (Ruan et al., 2024). VERSA focuses on safe task execution when shared tool interfaces conceal deployment-specific rules governing the consequences of tool actions.

2.2 World Models for Agents

World models predict action consequences to support planning, an approach rooted in model-based reinforcement learning (Sutton, 1991; Ha & Schmidhuber, 2018; Hafner et al., 2020; Chua et al., 2018). Model-based policy optimization (MBPO) uses short model-generated rollouts to improve policy learning (Janner et al., 2019). SafeDreamer incorporates safety constraints into world-model-based planning (Huang et al., 2024). For language agents, RAP uses an LLM as both a reasoning agent and a world model (Hao et al., 2023). WMA learns web transition dynamics (Chae et al., 2025), WebDreamer simulates web-action outcomes for planning (Gu et al., 2025), and ViMo predicts future visual observations for GUI agents (Luo et al., 2026).

A related line of work studies how agents acquire and maintain knowledge of their environments. Plan2Explore uses ensemble disagreement to guide exploration by planning with a learned world model (Sekar et al., 2020). Nair et al. (2026) investigate runtime discovery of enterprise execution logic through configuration inspection and targeted probing. EnvProbe uses budgeted environment probing to calibrate agent beliefs (Song & Cai, 2026), while Agent-BRACE represents state uncertainty through explicit verbalized beliefs (Singh et al., 2026). VERSA focuses on the rule ambiguity that can persist after gathering such evidence: multiple execution-rule configurations may remain plausible while predicting different outcomes for the same candidate plan.

Our approach builds on Bayesian inference over unknown environments (Ghavamzadeh et al., 2015; Guez et al., 2012) and hidden-parameter dynamics models (Doshi-Velez & Konidaris, 2016). PEARL and VariBAD infer latent task variables from experience to support adaptation and exploration (Rakelly et al., 2019; Zintgraf et al., 2019). Flow matching posterior estimation provides a generative approach to conditional inference (Wildberger et al., 2023). VERSA uses posterior inference to retain multiple deployment-specific rule configurations that explain the available evidence. A rule-conditioned transition model predicts plan outcomes under these hypotheses, supporting risk-constrained planning and information acquisition guided by decision value.

3 Method

VERSA represents uncertainty about deployment-specific execution rules through a joint posterior and predicts candidate-plan outcomes under the resulting rule hypotheses. These predictions guide risk-constrained plan selection and determine whether further information is worth acquiring. Figure 2 summarizes the framework: a conditional flow model infers rules, a rule-conditioned transition model predicts outcomes, and a controller chooses among execution, querying, and stopping.

3.1 Problem Formulation

We consider tool environments with shared interfaces and deployment-specific execution rules. A rule configuration $z^* \in \mathcal{Z}$ remains fixed within an episode and governs action preconditions, cascading effects, and the timing of state changes. Let s_t denote the environment state, including its clock and pending effects, and let o_t denote the observation available to the agent (Kaelbling et al., 1998). A tool action a_t induces

$$(s_{t+1}, o_{t+1}) \sim P_{z^*}(\cdot \mid s_t, a_t). \tag{1}$$

The agent observes tool responses and exposed state. We denote its observed interaction history by D_t and the configuration values returned by reads by R_t. Several complete rule configurations may explain the same evidence. VERSA represents this ambiguity through an approximate joint posterior $q_t(z)$, which provides a weighted representation of the version space.

A task specifies a goal g, safety invariants $\mathcal{I}$, and an initial budget B_0. At decision step t, the agent has remaining budget B_t and a given finite candidate set Π_t. Each candidate $\pi = (a_1, \ldots, a_{L_\pi})$ is a concrete tool-operation sequence. The controller can execute a candidate, issue a permitted information query, or choose π_{stop} to cease further intervention. Queries consume budget and advance time; stopping leaves existing pending effects to evolve through the episode's settlement window.

Let ξ be the complete episode trajectory, including queries, execution, and delayed-effect settlement. Define $G(\xi) \in \{0, 1\}$ as goal completion at the settled terminal state and $U(\xi) \in \{0, 1\}$ as any violation of $\mathcal{I}$ during the episode. Safe success is $S(\xi) = G(\xi)(1-U(\xi))$. Following the constrained

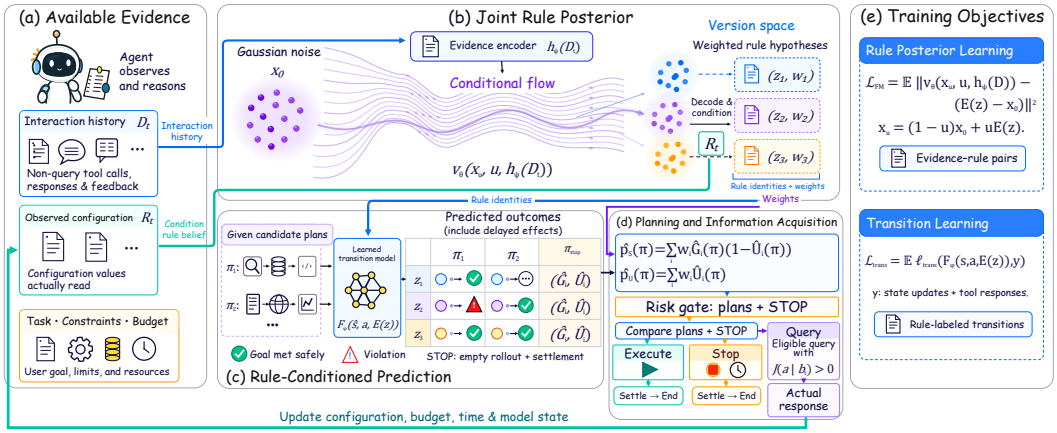


Figure 2: Overview of VERSA. (a) Interaction history and observed configuration provide deployment evidence. (b) A conditional flow model generates rule codes, which are decoded into hypotheses and conditioned on observed configuration. (c) A learned transition model predicts candidate-plan and STOP outcomes under each hypothesis, including delayed effects. (d) Weighted predictions guide execution and information acquisition; query responses update the evidence, budget, and time. (e) The two models learn from evidence–rule pairs and rule-labeled transitions.

formulation of safe reinforcement learning (Garcıa & Fernández, 2015; Achiam et al., 2017), we maximize safe success subject to a target violation probability r:

$$\max_{\mu} \mathbb{E}_\mu[S(\xi)] \qquad \text{subject to} \qquad \mathbb{E}_\mu[U(\xi)] \le r, \tag{2}$$

where μ selects permitted queries, candidate plans, or stopping within the available budget. Expectations cover deployments, tasks, environment outcomes, and policy randomness.

3.2 Learning a Joint Posterior over Rules

Evidence about one execution rule can change the plausibility of another. VERSA therefore learns a joint distribution over complete rule configurations to represent multiple plausible explanations and dependencies among their components.

Training supervision. We collect evidence–rule pairs (D, z) by executing tools in training deployments with known configurations. The observed history D is the input evidence, and the complete configuration z supplies supervision. We write $E(z) \in \mathbb{R}^d$ for a model's rule encoding. An evidence encoder $h_\psi(D)$ summarizes the non-query interaction records; configuration reads are incorporated separately as observed constraints.

Conditional flow matching. We learn the conditional distribution of rule codes using flow matching (Lipman et al., 2023; Tong et al., 2024; Wildberger et al., 2023). For $x_0 \sim \mathcal{N}(0, I)$ and interpolation time $u \sim \mathcal{U}[0, 1]$, define

$$x_1 = E(z), \qquad x_u = (1-u)x_0 + ux_1. \tag{3}$$

The velocity field is trained with

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{(D,z),x_0,u}\left[\|v_\theta(x_u, u, h_\psi(D)) - (x_1 - x_0)\|_2^2\right]. \tag{4}$$

At deployment, independent Gaussian initializations are transported by

$$\frac{dx(u)}{du} = v_\theta(x(u), u, h_\psi(D_t)), \qquad x(0) \sim \mathcal{N}(0, I). \tag{5}$$

The resulting endpoints are continuous rule representations, which are decoded into complete configurations. Here u indexes the generative flow, while t indexes interaction with the environment.

Rule hypotheses and observed configuration. In the catalogue implementation, each endpoint is decoded to the nearest encoded configuration in a public catalogue of legal rules. Sample frequencies yield a discrete distribution, calibrated on validation data, which we denote by $q_\theta(z \mid D_t)$. Let $\mathcal{C}(R_t)$ contain the configurations consistent with the values actually read. The current belief is

$$q_t(z) = \frac{q_\theta(z \mid D_t)\,\mathbf{1}\{z \in \mathcal{C}(R_t)\}}{\sum_{z'} q_\theta(z' \mid D_t)\,\mathbf{1}\{z' \in \mathcal{C}(R_t)\}}. \tag{6}$$

If conditioning removes all mass, we use the public prior restricted to $\mathcal{C}(R_t)$. New interaction records update the learned distribution, while exact configuration reads condition its rule probabilities. The resulting weighted hypotheses $\{(z_i, w_i)\}_{i=1}^K$, with $\sum_i w_i = 1$, provide the interface to consequence prediction. Each model uses its own encoding $E(z_i)$ of the shared rule identity.

3.3 Rule-Conditioned Transition Prediction

VERSA uses a learned transition model to evaluate candidate plans under different rule hypotheses. For each hypothesis, the model predicts the consequences of executing the same tool-operation sequence.

Learning action consequences. Tool execution in known-rule training deployments also provides tuples (s, a, z, y), where y contains the resulting structured state updates and tool response. State-update targets include when changes take effect. We train a rule-conditioned model F_ϕ with

$$\mathcal{L}_{\text{trans}} = \mathbb{E}_{(s,a,z,y)}\left[\ell_{\text{trans}}\big(F_\phi(s, a, E(z)), y\big)\right], \tag{7}$$

where ℓ_{trans} combines the supervised losses for the response and state-update fields. Given a rule hypothesis, the model produces structured predictions that are applied to advance the simulated state.

Predicting plan outcomes. For each hypothesis z_i, VERSA reconstructs a model state $\hat{s}_t^{(i)}$ from visible evidence and hypothesis-conditioned history replay. This state tracks predicted pending effects and the environment clock. Starting from $\hat{s}_t^{(i)}$, the model rolls out each $\pi \in \Pi_t$, keeping z_i fixed throughout the sequence. Goal and invariant predicates are applied to the predicted trajectory, including settlement, to obtain binary indicators $\widehat{G}_i(\pi)$ and $\widehat{U}_i(\pi)$. The violation indicator also includes violations in the reconstructed history. The same procedure evaluates π_{stop} as an empty operation sequence followed by settlement. These predictions allow the controller to compare executing a plan with stopping while pending effects settle.

3.4 Planning and Information Acquisition

The controller uses rule beliefs and predicted outcomes to choose among plan execution, information acquisition, and STOP. Let b_t collect the available evidence, weighted rule hypotheses, their model states, and the remaining budget.

Safe-success and risk estimates. For any candidate or STOP, VERSA computes

$$\begin{aligned} \widehat{p}_S(\pi \mid b_t) &= \sum_{i=1}^K w_i \widehat{G}_i(\pi)\big(1 - \widehat{U}_i(\pi)\big), \\ \widehat{p}_U(\pi \mid b_t) &= \sum_{i=1}^K w_i \widehat{U}_i(\pi). \end{aligned} \tag{8}$$

The first quantity estimates the probability of completing the task without violation; the second estimates violation probability. Let $\mathcal{C}_t = \{\pi \in \Pi_t : c_{\text{plan}}(\pi) \le B_t\} \cup \{\pi_{\text{stop}}\}$ contain all budget-feasible choices. STOP incurs no additional tool cost, and its predictions include the settlement of pending effects. All choices share the same risk gate:

$$\mathcal{A}_\kappa(b_t) = \{\pi \in \mathcal{C}_t : \widehat{p}_U(\pi \mid b_t) \le \kappa\}. \tag{9}$$

We select κ on validation data using empirical violation rates for the target risk level r. Set $\Omega(b_t) = \mathcal{A}_\kappa(b_t)$ when this set is nonempty; otherwise, set $\Omega(b_t) = \arg\min_{\pi \in \mathcal{C}_t} \widehat{p}_U(\pi \mid b_t)$ and record constraint infeasibility. The controller then maximizes predicted safe success:

$$\pi_t^* \in \arg\max_{\pi \in \Omega(b_t)} \widehat{p}_S(\pi \mid b_t), \qquad V(b_t) = \widehat{p}_S(\pi_t^* \mid b_t). \tag{10}$$

Ties favor lower nominal cost, then STOP and the fixed candidate order.

Valuing information. A query is useful when its answer improves the subsequent execution decision (Howard, 1966). For a configuration query a revealing a field $f_a(z)$, the predicted response probability is

$$\widehat{p}(o \mid a, b_t) = \sum_{i=1}^{K} w_i \, \mathbf{1}\{f_a(z_i) = o\}. \tag{11}$$

Each possible response induces a successor decision state $b_t^{a,o}$. This state incorporates the hypothetical returned value, the remaining budget $B_t - c(a)$, and the clock and model-state changes during the query. Thus, waiting for information can alter both the belief and the available plans.

Queries are eligible when permitted and affordable, and when their predicted query-then-stop violation probability is at most κ. Let $\mathcal{Q}(b_t)$ denote this eligible set. We score each query by its one-step expected improvement in safe success, less its acquisition cost:

$$J(a \mid b_t) = \sum_o \widehat{p}(o \mid a, b_t) V(b_t^{a,o}) - V(b_t) - \lambda c(a), \tag{12}$$

where $\lambda \geq 0$ sets the cost penalty. The same terminal selection rule defines V before the query and after each hypothetical response.

Execution loop. If an eligible query has a positive score, VERSA executes the highest-scoring query, incorporates its actual response, and reevaluates the decision. Otherwise, it selects π_t^* and commits to the chosen sequence or ceases further intervention; existing delayed effects are then allowed to settle. This decision rule allows the agent to stop acquiring information and proceed while uncertainty about some deployment rules remains.

4 Theoretical Analysis

We analyze how posterior and prediction errors affect execution-risk estimates and when identifying additional deployment rules offers little benefit. Condition on the current evidence (D_t, R_t). Let $p_t^*(z)$ be the true rule posterior and $\mathcal{C}_t$ the finite set of budget-feasible candidate plans, including STOP. For $X \in \{S, U\}$, let $f_X(\pi, z) \in [0, 1]$ denote the true conditional probability of outcome X, and $\widehat{f}_X(\pi, z)$ its model prediction, including history and settlement. Define

$$\alpha_t = \mathrm{TV}(p_t^*, q_t), \qquad \varepsilon_X = \mathbb{E}_{z \sim q_t}\left[\max_{\pi \in \mathcal{C}_t} \left| f_X(\pi, z) - \widehat{f}_X(\pi, z) \right|\right]. \tag{13}$$

Proposition 1 (Execution risk). For every $\pi \in \mathcal{C}_t$, its conditional violation probability satisfies

$$p_U^*(\pi) := \mathbb{E}_{z \sim p_t^*}[f_U(\pi, z)] \leq \widehat{p}_U(\pi \mid b_t) + \alpha_t + \varepsilon_U. \tag{14}$$

When $\mathcal{A}_\kappa(b_t) \neq \emptyset$, the selected choice π_t^*, including STOP, satisfies $p_U^*(\pi_t^*) \leq \kappa + \alpha_t + \varepsilon_U$.

The two error terms connect rule inference and consequence prediction to execution safety. The posterior error measures agreement with the evidence-conditioned rule distribution, which may itself contain multiple modes. Thus, preserving several plausible configurations is compatible with accurate risk estimation.

Remaining value of rule information. We compare the selected plan π_t^* with an oracle that observes the rule before choosing from $\mathcal{C}_t$. The comparison includes candidates that may become admissible after additional evidence. Writing $\widehat{S}_i(\pi) = \widehat{G}_i(\pi)(1 - \widehat{U}_i(\pi))$, define

$$\widehat{\Delta}_t = \sum_{i=1}^{K} w_i \left[\max_{\pi \in \mathcal{C}_t} \widehat{S}_i(\pi) - \widehat{S}_i(\pi_t^*)\right]. \tag{15}$$

This gap can be computed from the controller's plan predictions. It measures how much knowing the rule could improve the current choice. A hypothesis contributes zero when the selected plan is optimal under its predicted outcomes, even if other rules explain the environment differently.

Proposition 2 (Value of further information). Consider any adaptive sequence of rule observations that leaves execution conditions unchanged and ends with a plan in $\mathcal{C}_t$. Its expected improvement Δ_{info} in safe success over executing π_t^*, including the case of full rule revelation, satisfies

$$\Delta_{\text{info}} \leq \widehat{\Delta}_t + \alpha_t + 2\varepsilon_S. \tag{16}$$

If π_t^* is δ-optimal for predicted safe success on hypotheses of total weight at least $1 - \epsilon$, with $\epsilon, \delta \in [0, 1]$, then $\widehat{\Delta}_t \leq (1 - \epsilon)\delta + \epsilon$. The bound can be small even when many rule configurations remain plausible, provided their differences have little effect on the selected plan.

Corollary 1 (Decisions without full rule identification). If $\mathcal{A}_\kappa(b_t) \neq \emptyset$ and $\kappa + \alpha_t + \varepsilon_U \leq r$, the selected choice π_t^* has violation probability at most r and safe-success probability within $\eta_t = \widehat{\Delta}_t + \alpha_t + 2\varepsilon_S$ of the fully informed oracle over $\mathcal{C}_t$.

Together, these results characterize safe, near-optimal decisions through posterior accuracy and the remaining benefit of rule information. Further information-only acquisition has no positive net value when $\eta_t \leq \lambda c_{\min}$, where $c_{\min}$ is its minimum acquisition cost. For actual queries, Equation 12 also evaluates changes in budget, time, and state. Complete proofs and an extension to these query effects appear in Appendix A.

5 Experiments

5.1 Experimental Setup

Benchmarks. We construct retail and airline workflows from the public records and tool implementations of τ-bench (Yao et al., 2024). Deployments share tool interfaces but differ in action preconditions, refund cascades, effect timing, and failure reporting. We also evaluate task-composition transfer in the airline domain with frozen models and controller parameters, and examine posterior representations in a controlled settlement environment.

Baselines. We compare VERSA with ReAct (Yao et al., 2023), ReAct+RAG, WebDreamer (Gu et al., 2025), and RAP (Hao et al., 2023), using Qwen2.5-7B-Instruct (Yang et al., 2024) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) for the language-agent baselines. VERSA uses its own learned rule posterior and transition model. Benchmark construction, training details, and baseline adaptations are provided in Appendix B.

Metrics. We report goal success (G), safe success (SS), unsafe execution (U), and tool cost. G measures final task completion, U records whether any safety constraint is violated, and our primary metric SS requires completion without violations. Evaluation includes delayed effects; tool cost covers queries and execution. Public-workflow results average episodes within each source user and then weight users equally.

5.2 Experimental Results

Main Results. Table 1 shows that VERSA achieves the highest goal success and safe success and the lowest violation rate among the compared methods in both domains. Safe success reaches 97.92% on retail and 88.09% on airline, exceeding the strongest baseline in each domain by 38.54 and 21.86 percentage points, respectively. The substantial gap between goal success and safe success among language-agent baselines shows that reaching the task goal often comes with safety violations. VERSA narrows this gap while improving task completion under hidden deployment rules. It also achieves the lowest tool cost on retail.

Ablation Studies. Table 2 progressively adds three components to a learned world model without rule inputs. Rule-aware modeling alone uses the most probable inferred rule for prediction and

Table 1: Main results on retail and airline workflows. G: goal success; SS: safe success; U: unsafe execution. Rates are percentages. Best values are bold.

		Retail				Airline			
Backbone	Method	G↑	SS↑	U↓	Cost↓	G↑	SS↑	U↓	Cost↓
Qwen2.5 7B-Instruct	ReAct	73.44	52.86	22.14	10.26	71.37	56.01	32.60	5.67
	ReAct+RAG	67.97	47.92	20.83	10.91	74.54	56.39	25.43	5.68
	WebDreamer	84.38	59.11	26.04	10.87	86.32	64.98	24.72	5.02
	RAP	76.56	54.43	22.92	7.51	78.16	66.23	21.67	5.34
Llama-3.1 8B-Instruct	ReAct	84.11	54.43	29.69	12.00	77.39	55.44	25.60	5.91
	ReAct+RAG	72.40	52.60	20.31	12.00	74.96	49.01	34.16	5.95
	WebDreamer	80.73	56.25	27.08	12.00	83.59	55.61	33.15	5.98
	RAP	85.68	59.38	27.34	10.35	17.38	13.85	3.84	**1.30**
–	**VERSA**	**97.92**	**97.92**	**2.08**	**5.81**	**91.61**	**88.09**	**3.52**	3.20

Table 2: Progressive ablation on retail and airline workflows. SS and U denote safe success and unsafe execution (%). Best and second-best values are boldfaced and underlined, respectively.

Components			Retail			Airline		
Rule-aware world model	Version-space planning	Selective queries	SS↑	U↓	Cost↓	SS↑	U↓	Cost↓
×	×	×	53.65	22.14	<u>2.61</u>	33.95	<u>5.77</u>	**1.16**
✓	×	×	52.60	4.69	**2.39**	57.66	9.13	<u>1.69</u>
✓	✓	×	<u>94.01</u>	<u>3.91</u>	5.68	<u>87.03</u>	**3.52**	3.18
✓	✓	✓	**97.92**	**2.08**	5.81	**88.09**	**3.52**	3.20

planning. The largest gain comes from version-space planning, which increases safe success by 41.41 and 29.37 percentage points in retail and airline, respectively. This comparison keeps the Flow posterior and rule-conditioned transition model fixed and uses no additional queries, isolating the effect of considering multiple plausible rules when choosing a plan. Rule-aware modeling reduces retail violations and improves airline safe success over the rule-agnostic model. Selective queries provide a further improvement, particularly in retail, with little additional tool cost. Full VERSA achieves the highest safe success and the lowest or joint-lowest violation rate in both domains. Relative to full VERSA, simpler variants have lower tool costs and lower safe success.

Generalization Results. We test whether learned dynamics transfer to new task compositions.

We transfer VERSA from single-goal airline tasks to workflows in which two goals must both be achieved. The model weights and controller parameters remain fixed. Table 3 compares VERSA with the Qwen2.5-7B baselines using the same task budget. VERSA completes more tasks safely while incurring fewer violations and lower tool cost.

Table 3: Airline task transfer.

Method	SS (%)↑	U (%)↓	Cost↓
ReAct	66.75	19.10	9.18
ReAct+RAG	67.57	10.57	9.32
WebDreamer	67.57	25.58	8.63
VERSA	**87.75**	**0.16**	**5.49**

Safe success exceeds the strongest reported baseline by 20.18 percentage points.

5.3 Additional Analysis and Discussion

Execution Dynamics. We examine how execution timing affects safe task completion.

Figure 3 compares retail safe success under immediate execution, delayed execution, and asynchronous rollback. Delayed execution separates a tool response from the eventual state change, so pending effects can influence subsequent decisions. All four Qwen2.5-7B baselines achieve their lowest safe success under delayed execution, while VERSA safely completes over 94% of these tasks. These results support accounting for pending effects when planning under uncertain execution rules. Airline baselines show the same pattern (Appendix C.5).

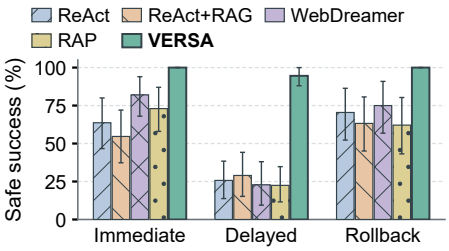


Figure 3: Retail safe success (95% CIs).

Posterior Representation and Efficiency. We evaluate rule inference under sparse evidence.

Table 4 compares four representations on the controlled benchmark with a shared transition model and controller. Distributional representations improve safe success and reduce violations relative to point estimates. Flow achieves the highest safe success and lowest violation rate, with per-episode runtime close to the full-covariance Gaussian. The safe-success ranking also holds under standard evidence (Appendix C.3).

Table 4: Posterior representations.

Posterior	SS↑	U↓	Time (s)↓
Point	69.53	25.78	**0.06**
Prior	85.55	2.34	1.73
Gaussian	87.11	2.34	1.65
Flow	**89.06**	**1.95**	1.67

Information Acquisition. Querying yields its largest gains under sparse evidence.

Figure 4 compares VERSA with its no-query variant across three levels of initial evidence on the controlled benchmark. VERSA achieves higher safe success throughout, showing that additional information remains useful even with richer histories. Replaying VERSA's queries while ignoring their answers reduces safe success (Appendix C.4). This control preserves query fees and elapsed time, linking the improvement to using acquired evidence to revise decisions.

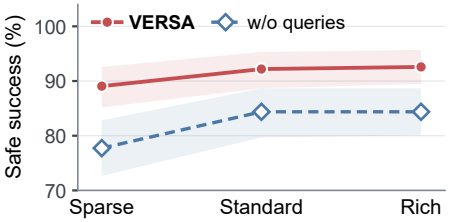


Figure 4: Initial evidence. Bands show 95% confidence intervals.

Parameter Analysis. We examine how the risk threshold shapes execution decisions.

Table 5 examines κ on the controlled benchmark with all other settings fixed. Strict thresholds reduce violations but also limit safe task completion. Relaxing the threshold improves safe success under both evidence conditions, with a gradual increase in violations. At $\kappa = 0.20$, VERSA reaches its highest safe success in the tested range while keeping violation rates below 3%.

Table 5: Effect of κ (rates in %).

	Sparse		Standard	
κ	SS↑	U↓	SS↑	U↓
0.00	76.17	0.00	80.08	0.00
0.02	83.20	0.00	86.33	0.00
0.05	85.16	0.39	88.28	0.00
0.10	87.11	1.17	91.41	0.78
0.20	89.06	1.95	92.19	2.73

6 Conclusion

We introduced VERSA, a world model that represents plausible execution rules as a version space. Its learned joint rule posterior and rule-conditioned transition model support risk-constrained planning and selective information acquisition. Experiments on retail and airline workflows show improved safe task completion, while controlled analyses support the benefit of planning across multiple rule hypotheses. Our theoretical results characterize conditions for safe, near-optimal decisions over the candidate set without fully identifying deployment rules.

AI USE STATEMENT

We used generative AI tools to assist with language refinement. These tools also supported literature searches to identify studies related to our research. We used AI tools to assist with the execution of experiments.

REPRODUCIBILITY STATEMENT

The source code is provided in the supplementary material. Appendix A contains the theoretical proofs, and Appendix B describes the benchmark construction and experimental settings.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. Pmlr, 2017.

Hyungjoo Chae, Namyoung Kim, Kai Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, volume 2025, pp. 63707–63738, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, pp. 1432, 2016.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. *Advances in Neural Information Processing Systems*, 23, 2010.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=c6l7yA0HSq.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. *Advances in neural information processing systems*, 25, 2012.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

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.

Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.

Tairan Huang, Siyu Shang, Qiang Chen, Xiu Su, and Yi Chen. Tools as continuous flow for evolving agentic reasoning. *arXiv preprint arXiv:2605.07339*, 2026.

Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. Safedreamer: Safe reinforcement learning with world models. In *International Conference on Learning Representations*, volume 2024, pp. 53839–53869, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *Artificial Intelligence and Statistics*, pp. 430–438. PMLR, 2014.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. Vimo: A generative visual gui world model for app agents. In *International Conference on Learning Representations*, volume 2026, pp. 117437–117463, 2026.

Tom M Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pp. 305–310, 1977.

Jishnu Sethumadhavan Nair, Patrice Bechard, Rishabh Maheshwary, Surajit Dasgupta, Sravan Ramachandran, Aakash Bhagat, Shruthan Radhakrishna, Pulkit Pattnaik, Johan Obando-Ceron, Shiva Krishna Reddy Malay, et al. Do enterprise systems need learned world models? the importance of context to infer dynamics. *arXiv preprint arXiv:2605.12178*, 2026.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024.

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

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pp. 8583–8592. PMLR, 2020.

Joykirat Singh, Zaid Khan, Archiki Prasad, Justin Chih-Yao Chen, Akshay Nambi, Hyunji Lee, Elias Stengel-Eskin, and Mohit Bansal. Agent-brace: Decoupling beliefs from actions in long-horizon tasks via verbalized state uncertainty. *arXiv preprint arXiv:2605.11436*, 2026.

Xinyuan Song and Zekun Cai. Ask the world before acting: Budgeted environment probing for world-model calibration. *arXiv preprint arXiv:2606.31422*, 2026.

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

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=CD9Snc73AW.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. *Advances in Neural Information Processing Systems*, 36:16837–16864, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

A PROOFS

This appendix proves the results in Section 4. We condition throughout on the current evidence, task, budget, and realized model predictions, and use the notation of the main text. In particular, $\mathcal{C}_t$ contains the budget-feasible candidate plans and STOP, and $q_t = \sum_{i=1}^{K} w_i \delta_{z_i}$ is the distribution used by the controller. All outcome probabilities include history and delayed-effect settlement. We use the total variation convention $\mathrm{TV}(p, q) = \frac{1}{2} \sum_z |p(z) - q(z)|$. For any $h : \mathcal{Z} \to [0, 1]$, this gives

$$\left|\mathbb{E}_{p_t^*}[h(z)] - \mathbb{E}_{q_t}[h(z)]\right| \leq \mathrm{TV}(p_t^*, q_t) = \alpha_t. \tag{17}$$

A.1 PROOF OF PROPOSITION 1

For any $\pi \in \mathcal{C}_t$, adding and subtracting its violation probability under q_t gives

$$\begin{aligned} p_U^*(\pi) - \widehat{p}_U(\pi \mid b_t) &= \left(\mathbb{E}_{p_t^*} - \mathbb{E}_{q_t}\right)[f_U(\pi, z)] \\ &\quad + \mathbb{E}_{q_t}[f_U(\pi, z) - \widehat{f}_U(\pi, z)] \\ &\leq \alpha_t + \mathbb{E}_{q_t}\left[\left|f_U(\pi, z) - \widehat{f}_U(\pi, z)\right|\right] \\ &\leq \alpha_t + \varepsilon_U. \end{aligned} \tag{18}$$

The first inequality follows from Equation 17, since $f_U(\pi, z) \in [0, 1]$. The second follows from the definition of ε_U in Equation 13. Therefore,

$$p_U^*(\pi) \leq \widehat{p}_U(\pi \mid b_t) + \alpha_t + \varepsilon_U. \tag{19}$$

This holds simultaneously for all candidates, including a plan selected using the model predictions. When $\mathcal{A}_\kappa(b_t) \neq \emptyset$, the controller selects $\pi_t^* \in \mathcal{A}_\kappa(b_t)$. This includes STOP, whose outcome predictions account for the settlement of pending effects. Substituting the gate condition into the bound above gives

$$p_U^*(\pi_t^*) \leq \kappa + \alpha_t + \varepsilon_U.$$

When $\mathcal{A}_\kappa(b_t) = \emptyset$, the controller selects a choice with minimum predicted risk over $\mathcal{C}_t$. The same general bound gives

$$p_U^*(\pi_t^*) \leq \min_{\pi \in \mathcal{C}_t} \widehat{p}_U(\pi \mid b_t) + \alpha_t + \varepsilon_U.$$

This fallback is recorded as constraint infeasibility, since no available terminal choice passes the predictive risk gate.

These bounds condition on the current evidence and apply to committing to a terminal choice. The query-then-STOP eligibility check evaluates that specific continuation; it does not by itself establish a risk bound for the complete adaptive query policy.

In the experiments, κ is selected using empirical violation rates on development data. The sufficient condition $\kappa + \alpha_t + \varepsilon_U \leq r$ additionally requires control of posterior and prediction errors; empirical threshold selection alone does not establish this condition.

A.2 PROOF OF PROPOSITION 2

Fix the currently selected plan π_t^* and define

$$\begin{aligned} m(z) &= \max_{\pi \in \mathcal{C}_t} f_S(\pi, z), \quad g(z) = m(z) - f_S(\pi_t^*, z), \\ \widehat{m}(z) &= \max_{\pi \in \mathcal{C}_t} \widehat{f}_S(\pi, z), \quad \widehat{g}(z) = \widehat{m}(z) - \widehat{f}_S(\pi_t^*, z). \end{aligned} \tag{20}$$

Because $\pi_t^* \in \mathcal{C}_t$ and all outcome probabilities lie in $[0, 1]$, both g and $\widehat{g}$ lie in $[0, 1]$. Moreover, $\widehat{\Delta}_t = \mathbb{E}_{q_t}[\widehat{g}(z)]$.

Consider an adaptive rule-observation strategy ν that terminates with a plan $\pi_H \in \mathcal{C}_t$, where H denotes its observation history, including any internal randomization. Under the proposition's information-only setting, observations depend only on the rule and previous observation history, and leave execution conditions unchanged. They reveal no additional state or outcome randomness

conditional on the rule. Thus, conditional on z and H, the expected safe success of the selected plan is $f_S(\pi_H, z)$. For every z and H, $f_S(\pi_H, z) \le m(z)$, so

$$\begin{aligned}\Delta_{\text{info}}(\nu) &= \mathbb{E}_{z\sim p_t^*,\, H|z,\nu}[f_S(\pi_H, z)] - \mathbb{E}_{p_t^*}[f_S(\pi_t^*, z)] \\ &\le \mathbb{E}_{p_t^*}[g(z)]. \end{aligned} \tag{21}$$

Full rule revelation allows the oracle to attain $\mathbb{E}_{p_t^*}[m(z)]$ by choosing the best plan in $\mathcal{C}_t$ for each rule.

To relate the true gap to the model predictions, write

$$e_S(z) = \max_{\pi\in\mathcal{C}_t} \left| f_S(\pi, z) - \widehat{f}_S(\pi, z) \right|. \tag{22}$$

The inequality $|\max_\pi a_\pi - \max_\pi b_\pi| \le \max_\pi |a_\pi - b_\pi|$ gives

$$|m(z) - \widehat{m}(z)| \le e_S(z), \qquad |g(z) - \widehat{g}(z)| \le 2e_S(z). \tag{23}$$

Applying Equation 17 to g yields

$$\begin{aligned}\mathbb{E}_{p_t^*}[g(z)] &\le \mathbb{E}_{q_t}[g(z)] + \alpha_t \\ &\le \mathbb{E}_{q_t}[\widehat{g}(z)] + 2\mathbb{E}_{q_t}[e_S(z)] + \alpha_t \\ &= \widehat{\Delta}_t + \alpha_t + 2\varepsilon_S. \end{aligned} \tag{24}$$

Together with Equation 21, this proves the claimed bound for any such observation strategy, including full rule revelation.

Finally, suppose π_t^* is δ-optimal under the model predictions on a set $\mathcal{Z}_0$ with $q_t(\mathcal{Z}_0) \ge 1-\epsilon$. Then $\widehat{g}(z) \le \delta$ on $\mathcal{Z}_0$ and $\widehat{g}(z) \le 1$ elsewhere. Writing $a = q_t(\mathcal{Z}_0^c) \le \epsilon$, we obtain

$$\widehat{\Delta}_t \le (1-a)\delta + a \le (1-\epsilon)\delta + \epsilon. \tag{25}$$

Thus, the remaining value of information can be small even when several rule configurations retain posterior mass.

A.3 Proof of Corollary 1

When $\mathcal{A}_\kappa(b_t) \ne \emptyset$, the selection rule in Equation 10 gives $\pi_t^* \in \mathcal{A}_\kappa(b_t)$. Proposition 1 therefore gives

$$p_U^*(\pi_t^*) \le \kappa + \alpha_t + \varepsilon_U \le r. \tag{26}$$

The fully informed oracle's advantage over π_t^* is

$$\begin{aligned}&\mathbb{E}_{p_t^*}[m(z)] - \mathbb{E}_{p_t^*}[f_S(\pi_t^*, z)] \\ &\quad = \mathbb{E}_{p_t^*}[g(z)] \le \widehat{\Delta}_t + \alpha_t + 2\varepsilon_S = \eta_t, \end{aligned} \tag{27}$$

where the inequality follows from Equation 24. These two inequalities establish the risk guarantee and the safe-success gap stated in the corollary.

Let C_ν denote the total acquisition cost of an information-only strategy ν. If $\mathbb{E}[C_\nu] \ge c_{\min}$, its net improvement satisfies

$$\Delta_{\text{info}}(\nu) - \lambda\mathbb{E}[C_\nu] \le \eta_t - \lambda c_{\min}. \tag{28}$$

Hence, if every strategy that acquires further rule information has expected cost at least $c_{\min}$, $\eta_t \le \lambda c_{\min}$ is sufficient for further information-only acquisition to have no positive net value.

Queries that change execution conditions. For an actual query a and response o, let $f_S^{a,o}(\pi, z)$ denote safe-success probability after the query followed by plan π, including query-time state changes and settlement. Suppose the eventual plan $\pi_{a,o}$ remains in $\mathcal{C}_t$, and let $\beta_t(a) \ge 0$ uniformly bound the positive change $f_S^{a,o}(\pi, z) - f_S(\pi, z)$ over possible rules, responses, and feasible successor plans. Then $f_S^{a,o}(\pi_{a,o}, z) \le m(z) + \beta_t(a)$, which gives

$$\begin{aligned}&\mathbb{E}_{z\sim p_t^*,\, o|z,a}[f_S^{a,o}(\pi_{a,o}, z)] - \mathbb{E}_{p_t^*}[f_S(\pi_t^*, z)] - \lambda c(a) \\ &\quad \le \eta_t + \beta_t(a) - \lambda c(a). \end{aligned} \tag{29}$$

Here budget consumption restricts the available successor plans, while $\beta_t(a)$ accounts for potential benefits from changed execution conditions, such as waiting for pending effects. The controller evaluates these query effects through the successor states in Equation 12.

B Experimental Details

B.1 Benchmark Construction

Data sources and workflow construction. We construct retail and airline workflows from the public business records and tool implementations of τ-bench (Yao et al., 2024). Each workflow is grounded in a source user's records, including orders and products in retail, or reservations and flights in airline. We build tasks and deployment-specific execution rules on top of these resources. Retail tasks combine the cancellation of one order with the return and refund of another order belonging to the same user. Airline tasks involve reservation cancellation and refund, cabin changes, or refund processing. These workflows require agents to coordinate operations whose effects depend on the current business state and deployment rules.

Deployment rules and execution. The rule layer varies action preconditions, refund cascades, effect timing, and failure reporting while retaining shared tool interfaces. For example, cancellation may issue a refund automatically or require a separate refund operation. A tool response may precede the corresponding state change, and an operation may subsequently be rolled back. These variations affect which action sequences complete the task safely: issuing another refund while an earlier refund is pending can produce a duplicate transaction. Each episode uses a fixed rule configuration and begins with a tool-interaction history generated under that configuration. Agents receive the task, public observations, tool descriptions, and possible rule meanings, and may acquire additional configuration information during execution. Tool operations are executed in the rule-adapted environment, with pending effects settled before final goal evaluation.

Evaluation instances and task composition. Training and evaluation users are separated at the source-user level, with task and deployment variants kept within the same user's partition. We construct evaluation instances with an unfinished goal, a violation-free initial history, and a budget-feasible safe candidate route. The retail evaluation contains 122 episodes from 32 users, and the single-goal airline evaluation contains 574 episodes from 128 users. To evaluate task-composition transfer, we additionally construct airline workflows that combine two reservation goals and require both to be achieved. This evaluation contains 618 episodes from 128 users drawn from the same test-user pool. VERSA uses the single-goal model weights and controller parameters without further training or tuning for the composite tasks.

Controlled settlement environment. We also construct a batch-settlement environment for the posterior, information-acquisition, and parameter analyses. Deployments differ in hidden setup and per-item processing times, which determine when submitted payments settle. The agent must coordinate payment submission, waiting, verification, and shipping; shipping before settlement incurs an irreversible safety violation. Historical sandbox trials provide evidence about the hidden processing parameters, while additional probes and receipt checks consume execution time and budget. We evaluate 256 held-out deployments under Sparse, Standard, and Rich conditions, exposing the first 2, 6, and 8 observations from the same deployment histories, respectively. Posterior comparisons use a shared transition model and controller to examine how alternative rule representations affect execution decisions.

B.2 Baseline Implementations

We evaluate four language-agent baselines using the original instruction-tuned checkpoints of Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct. The agents receive task descriptions, public interaction histories, and the available tools and candidate plans. WebDreamer and RAP are adapted to the retail and airline tool interfaces, with the selected backbone also serving as their world model and value evaluator.

ReAct. ReAct (Yao et al., 2023) interleaves reasoning with action execution. At each step, it selects an operation from the current observation and interaction history, then incorporates the returned feedback into its next decision. We use a decoding temperature of zero and a maximum of 320 output tokens per response.

ReAct+RAG. This baseline augments ReAct with retrieved experience from the corresponding domain's training partition. A BM25 retriever selects the three most relevant examples and includes them in the action-selection prompt. The agent otherwise follows the same interaction and decoding procedure as ReAct.

WebDreamer. Our adaptation of WebDreamer (Gu et al., 2025) proposes three candidate actions and uses the LLM world model to generate two imagined one-step outcomes for each candidate. The LLM scores these outcomes, and the mean score determines the selected action. After executing that action, the agent replans using the actual environment feedback.

RAP. Our adaptation of RAP (Hao et al., 2023) uses the LLM to predict action consequences and estimate their value within Monte Carlo tree search. The search proposes up to three candidates per expansion, uses a maximum imagined depth of six, and sets the UCT exploration coefficient to 1.0. Following initialization, it performs eight simulations at the first decision and twelve at subsequent decisions. The agent executes the most-visited root action and continues planning from the resulting observation.

B.3 Evaluation Metrics Formulation

Goal Success (G). The task goal is satisfied after delayed effects settle. Composite tasks require all constituent goals to be achieved.

Safe Success (SS). The task is completed without any safety violation, defined as $\mathrm{SS} = G(1 - U)$ for each episode.

Unsafe Execution (U). At least one specified safety invariant is violated during the episode, including delayed-effect settlement.

Tool Cost. The total charged cost of additional queries and task-execution operations.

Public-workflow metrics are averaged within each source user and then equally across users. Controlled metrics are averaged over deployments. Reported confidence intervals use bootstrap resampling of the corresponding users or deployments.

B.4 Implementation Details

Models and training. Each domain uses a separately trained rule posterior and transition model, whose weights remain fixed during evaluation. The posterior learns from paired interaction histories and rules; the transition model predicts tool responses and state updates from state–action–rule tuples. In retail, the evidence encoder is a four-layer, six-head Transformer with hidden dimension 384. The rule representation has dimension 32, the flow network has hidden width 512, and the transition decoder is a residual MLP of width 768 with response, value, effect-type, and delay prediction heads. Starting from previously trained checkpoints, both retail modules undergo 8,000 adaptation steps on data from training users with AdamW, a learning rate of 3×10^{-4}, 200 warmup steps, and cosine decay. Checkpoint selection uses validation loss; inference uses exponential moving averages of the model weights with decay 0.999.

Posterior inference. Retail inference transports 256 Gaussian samples through eight Euler steps and decodes the endpoints to the nearest rule representations in the 36-configuration catalogue. The resulting frequencies are temperature-scaled and mixed with a uniform distribution before conditioning on observed configuration reads. Inference settings are selected on development users with an empirical violation target of $r = 0.05$.

Candidate plans. Plans are constructed from public operation templates and entities visible in the task. For composite tasks, routines for the individual goals are combined and duplicate sequences are removed. The retail panel contains 86–106 candidates per instance, each with two to eight operations. Public guards determine whether a step is skipped based on visible state, and nominal plan costs are checked against the remaining budget. The candidate menu is available to every method.

The true deployment rule is used separately to check instance solvability and score execution; it is not an input to candidate generation.

B.5 CONFIGURATIONS

Models are implemented in PyTorch. Language-agent evaluation uses vLLM 0.8.5 with FP16 inference on NVIDIA V100 GPUs with 32 GB memory. The predictive risk thresholds are $\kappa = 0.30$ for retail and $\kappa = 0.20$ for airline, with query-cost coefficient $\lambda = 0.01$. The controlled benchmark uses $\kappa = 0.20$ by default.

C ADDITIONAL EXPERIMENTS

C.1 POSTERIOR QUALITY ON CONSTRUCTED TASKS

We evaluate rule-posterior quality on a constructed benchmark spanning four domains, with 688 episodes derived from 86 base tasks. Evaluation uses new task instances within held-out deployment schemas. All methods receive the same evidence and a catalog of 36 rules. Flow and diagonal Gaussian posteriors use empirical distributions obtained from 64 samples projected onto this catalog. For scoring, we apply $\tilde{q}_\ell = (1 - \epsilon)q_\ell + \epsilon/36$, with $\epsilon = 10^{-3}$. Scores are averaged equally over domains, tasks within each domain, and rule variants within each task.

Table 6: Rule-posterior quality on constructed tasks. Brackets denote 95% confidence intervals from 2,000 task-cluster bootstrap resamples stratified by domain. Log score is the negative log probability of the true rule, in nats. Lower is better; best point estimates are boldfaced.

Posterior	Joint Brier↓	Log score↓
Diagonal Gaussian	1.068 [1.034, 1.102]	4.502 [4.224, 4.778]
Factorized categorical	0.932 [0.909, 0.957]	3.111 [2.806, 3.459]
Joint categorical	1.054 [1.025, 1.085]	4.255 [3.964, 4.560]
Autoregressive categorical	1.005 [0.976, 1.035]	3.637 [3.307, 3.992]
Qwen2.5-7B (prompted)	1.822 [1.805, 1.838]	9.672 [9.585, 9.753]
Qwen2.5-32B + RAG (prompted)	1.329 [1.245, 1.414]	7.441 [6.882, 7.947]
Evidence-updated enumeration	0.915 [0.904, 0.927]	**2.679** [2.550, 2.821]
Static uniform prior	0.972 [0.972, 0.972]	3.584 [3.584, 3.584]
Flow	**0.911** [0.887, 0.935]	3.049 [2.709, 3.420]

Table 6 reports joint Brier and negative log scores. Flow attains the lowest Brier score and improves both scores over the Gaussian and prompted LLM posteriors, while evidence-updated enumeration achieves the lowest negative log score.

C.2 POSTERIOR COMPARISON UNDER A SHARED CONTROLLER

We examine how posterior estimation affects downstream decisions by replacing the rule posterior while keeping the transition model, candidate plans, query mechanism, and safety controller fixed. Learned-likelihood enumeration scores all legal rules using approximate history likelihoods from the shared transition model. Joint categorical prediction outputs a distribution over the same rule catalogue. The airline classifier uses the Flow evidence encoder for initialization, with its checkpoint selected by held-out validation NLL. Each posterior is calibrated on the same development split using identical search grids and selection criteria. This comparison uses separate calibration settings; the main results retain their original configurations.

We evaluate 122 retail episodes from 32 source users and 574 airline episodes from 128 source users. Metrics are averaged within each user and then equally across users. Table 7 reports the results. Flow achieves the highest or joint-highest safe success in both domains. Enumeration matches its retail outcomes with lower tool cost. In the airline domain, Flow has the lowest tool cost, while joint classification achieves the lowest violation rate.

Table 7: Posterior comparison under a shared controller. SS and U are percentages; Cost denotes average tool fees. Best point estimates are boldfaced.

	Retail			Airline		
Posterior	SS↑	U↓	Cost↓	SS↑	U↓	Cost↓
Learned-likelihood enumeration	**97.92**	**2.08**	**5.315**	87.97	2.34	3.490
Joint categorical	95.57	3.13	5.693	87.42	**0.52**	3.503
Flow matching	**97.92**	**2.08**	5.609	**88.09**	3.52	**3.200**

Table 8: Paired differences (Flow minus comparator) with 95% confidence intervals. SS and U differences are in percentage points; cost differences are in tool-fee units.

Domain	Comparator	ΔSS	ΔU	ΔCost
Retail	Enumeration	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	+0.294 [0.047, 0.555]
Retail	Joint categorical	+2.34 [0.00, 5.47]	−1.04 [−3.65, 1.04]	−0.083 [−0.255, 0.070]
Airline	Enumeration	+0.12 [−1.82, 2.25]	+1.17 [−0.78, 3.91]	−0.290 [−0.409, −0.166]
Airline	Joint categorical	+0.67 [−0.35, 1.95]	+2.99 [0.65, 5.99]	−0.303 [−0.424, −0.180]

Table 8 reports paired differences with 95% confidence intervals from 2,000 bootstrap resamples of source users. The paired intervals include zero for the safe-success differences, while identifying differences in tool cost and airline violations. Together with the component ablation, these comparisons show that high safe success is retained across posterior implementations, with distinct trade-offs in safety and tool cost.

C.3 Posterior Representation and Efficiency

We extend the posterior comparison to both evidence conditions on the continuous-rule controlled benchmark. Sparse and Standard provide two and six initial observations, respectively, on the same 256 deployments. All representations share the transition model and controller; the Gaussian uses a full covariance matrix.

Table 9: Posterior representations under sparse and standard evidence. SS and U are percentages; Time is mean controller runtime per episode in seconds. Best values are boldfaced.

	Sparse			Standard		
Posterior	SS↑	U↓	Time (s)↓	SS↑	U↓	Time (s)↓
Point	69.53	25.78	**0.06**	74.22	24.61	**0.05**
Prior	85.55	2.34	1.73	85.94	**1.95**	1.71
Gaussian	87.11	2.34	1.65	90.63	2.73	1.48
Flow	**89.06**	**1.95**	1.67	**92.19**	2.73	1.50

Table 9 shows that distributional representations improve safe success and reduce violations relative to point estimates. Flow achieves the highest safe success in both conditions, with runtime close to that of the Gaussian.

C.4 Information Acquisition

We examine how acquired information affects decisions on the same controlled benchmark. The matched-query MAP variant replays VERSA's probes but plans using the most probable rule hypothesis. The no-query variant removes active probes while retaining ordinary status checks, and the fixed strategy issues one initial probe. The answer-ignored control replays VERSA's probes with the same fees and timing but discards their answers. Equal-time waiting replaces these probes with waits of equal duration, without query fees.

Table 10: Information-acquisition controls. SS and U are percentages; Q is the mean number of active queries per episode. Best SS and U values are boldfaced.

	Sparse			Standard		
Variant	SS↑	U↓	Q	SS↑	U↓	Q
MAP with matched queries	70.70	27.73	0.53	76.56	21.88	0.34
No queries	77.73	2.73	0.00	84.38	3.52	0.00
Fixed initial query	76.95	**1.56**	1.00	79.30	**0.78**	1.00
Query answers ignored	72.66	**1.56**	0.53	82.42	3.13	0.34
Equal-time waiting	75.00	**1.56**	0.00	82.81	3.13	0.00
VERSA	**89.06**	1.95	0.53	**92.19**	2.73	0.34

Table 10 shows that VERSA achieves the highest safe success in both conditions. The gains over matched-query MAP support retaining multiple hypotheses during planning. The answer-ignored and waiting controls further show that using query responses contributes beyond the elapsed time alone.

C.5 Execution Dynamics

Table 11 compares execution mechanisms in Retail and Airline. All four LLM baselines achieve their lowest safe success under delayed execution. VERSA attains 94.57% and 92.53% in this regime, respectively, supporting the value of anticipating pending effects.

Table 11: Safe success (%) across execution mechanisms; higher is better. LLM baselines use Qwen2.5-7B. Results give equal weight to users within each stratum. Best values are boldfaced.

	Retail			Airline		
Method	Immediate	Delayed	Rollback	Immediate	Delayed	Rollback
ReAct	63.67	25.72	70.45	85.69	34.27	67.35
ReAct+RAG	54.67	28.99	63.26	78.61	39.30	69.39
WebDreamer	82.00	22.83	75.00	90.24	44.91	77.55
RAP	73.00	22.46	62.12	**91.85**	44.32	**85.71**
VERSA	**100.00**	**94.57**	**100.00**	90.13	**92.53**	**85.71**