When LangChain Agents Collapse: Inference Failure and Bayesian Recovery

Angshul Majumdar

Abstract

LangChain-style tool-augmented language model agents are typically engineered as deterministic pipelines with validation-and-retry loops. While effective in many settings, these designs often exhibit brittle behavior: once a system commits to an early tool choice or intermediate structured output, subsequent steps can inherit that commitment even when better alternatives exist.

This paper reframes LangChain agent execution as sequential inference over action trajectories under a finite-horizon generative model of actions and tool observations. Within this unified view, deterministic greedy agents, probabilistic forward-sampling agents, and Bayesian particle-based agents correspond to distinct inference regimes over the same underlying interaction contract. Retry-based execution can be interpreted as conditional sampling under binary acceptance, whereas Bayesian Sequential Monte Carlo maintains multiple competing hypotheses and updates their plausibility using likelihood signals derived from tool outputs (e.g., schema validity, reliability, or semantic consistency).

Our goal is conceptual rather than benchmark-driven: we provide a clean inference lens for understanding why collapse occurs in common agent pipelines, and a principled recovery mechanism that preserves alternative trajectories long enough for evidence to accumulate. The accompanying reference implementations instantiate this inference ladder within the LangChain setting while keeping tool interfaces unchanged.

1. Introduction

Tool-augmented language model agents have rapidly emerged as a dominant paradigm for building interactive AI systems. In such systems, a large language model (LLM) alternates between natural language reasoning and

external tool calls: retrieving documents, executing code, querying APIs, or invoking domain-specific functions. The resulting interaction forms a sequential action–observation loop, where each tool invocation produces an observation that influences subsequent decisions.

Despite their practical success, most existing agent frameworks implement tool selection through deterministic parsing and greedy action choice, with heuristic retry mechanisms handling failure. When a tool call produces malformed output or an unexpected response, the system simply retries. When an early decision turns out to be suboptimal, downstream reasoning proceeds along the chosen branch without reconsideration of alternatives.

These behaviors are not incidental engineering artifacts. They are structural.

In this paper, we argue that current tool-augmented LLM agents implicitly perform greedy maximum a posteriori (MAP) inference over action sequences, collapsing uncertainty at each step. Retry loops act as a crude form of rejection sampling over proposed actions, but do not maintain posterior uncertainty over competing trajectories. As a result, early mis-specifications cannot be recovered, and stochastic tool outputs are not handled in a principled probabilistic manner.

Yet the problem agents solve is inherently stochastic. Language model outputs are probabilistic. Tool responses may be noisy, partially structured, or ambiguous. User intent is often underdetermined. From a decision-theoretic perspective, tool-augmented agents operate in a partially observable, sequential environment. Maintaining uncertainty over action trajectories is therefore not an optional refinement—it is the natural formulation of the problem.

1.1. From Greedy Selection to Posterior Reasoning

We formalize tool-augmented LLM agents through a simple generative model. Let $a_{1:T}$ denote a sequence of tool-selection actions over a finite horizon T, and let $o_{1:T}$ denote the corresponding observations produced by tool invocations. Writing $h_t = (a_{<t}, o_{<t})$ for the interaction history, we assume a factorization of the joint distribution

$$p(a_{1:T}, o_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t)\, p(o_t \mid a_t, h_t). \tag{1.1}$$

Here $p(a_t \mid h_t)$ represents the LLM's action proposal distribution condi-

tioned on the interaction history, and $p(o_t \mid a_t)$ models the stochastic tool response.

Under this model, the inferential objective is naturally the posterior distribution

$$p(a_{1:T} \mid o_{1:T}), \tag{1.2}$$

which captures uncertainty over complete action trajectories given observed tool outcomes.

Current deterministic agent implementations correspond to a sequential greedy approximation:

$$a_t = \arg\max_a p(a \mid h_t), \tag{1.3}$$

with optional retries if observations violate formatting constraints. This procedure collapses the trajectory distribution to a single path. No posterior mass is retained on alternative branches once a decision has been made.

1.2. An Inference Ladder for LLM Agents

We introduce a unified probabilistic view of agent inference that admits three regimes under the same generative model:

- **Deterministic Greedy Inference:** Sequential MAP selection with no uncertainty retention.
- **Probabilistic Sampling:** Forward sampling from $p(a_t \mid h_t)$ without posterior weighting.
- **Bayesian Sequential Monte Carlo (SMC):** Approximate posterior inference over action trajectories using weighted particles.

In the Bayesian formulation, a set of particles $\{a_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ approximates the posterior $p(a_{1:t} \mid o_{1:t})$. Particle weights evolve according to

$$w_t^{(i)} \propto w_{t-1}^{(i)}\, p(o_t \mid a_t^{(i)}, h_t^{(i)}), \tag{1.4}$$

with resampling maintaining support on high-likelihood trajectories. Unlike greedy selection, this framework preserves uncertainty over competing action prefixes and permits recovery from early suboptimal proposals when later evidence favors alternative paths.

1.3. Contributions

This paper makes the following contributions:

- We formalize tool-augmented LLM agents as sequential probabilistic models with explicit action and observation structure.
- We show that deterministic agent frameworks correspond to greedy MAP inference under this model.
- We interpret retry mechanisms as degenerate rejection sampling that does not perform posterior weighting.
- We propose a Bayesian Sequential Monte Carlo formulation that maintains uncertainty over action trajectories.
- We provide theoretical results (in the appendix) characterizing brittleness under greedy collapse and support preservation under particle-based inference.

Our goal is not to replace existing agent frameworks, but to generalize them. By making the underlying inference assumptions explicit, we provide a principled probabilistic foundation for uncertainty-aware tool-augmented LLM agents.

2. Formal Model of Tool-Augmented Agents

We now formalize tool-augmented language model agents under a unified probabilistic framework. Throughout, we assume finite spaces to avoid measure-theoretic complications and to make all statements precise.

2.1. Basic Definitions

Let $\mathcal{A}$ denote a finite action space of tool calls. An element $a_t \in \mathcal{A}$ represents the action selected at interaction step t.

Let $\mathcal{O}$ denote a finite observation space. An element $o_t \in \mathcal{O}$ represents the observation returned by the tool upon execution of action a_t.

We consider a finite interaction horizon $T \in \mathbb{N}$. An action trajectory is denoted

$$a_{1:T} = (a_1, \ldots, a_T) \in \mathcal{A}^T,$$

and the corresponding observation sequence is

$$o_{1:T} = (o_1, \ldots, o_T) \in \mathcal{O}^T.$$

At time t, the interaction history is defined as

$$h_t = (a_{<t}, o_{<t}) = (a_1, \ldots, a_{t-1}, o_1, \ldots, o_{t-1}).$$

2.2. Generative Model

We assume the following generative factorization of the joint distribution:

$$p(a_{1:T}, o_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t)\, p(o_t \mid a_t). \tag{2.1}$$

The two components are interpreted as follows:

- $p(a_t \mid h_t)$ is the LLM-induced action proposal distribution, conditioned on the full interaction history.
- $p(o_t \mid a_t, h_t)$ is the tool likelihood model, specifying the stochastic response of the tool to action a_t under the interaction history h_t.

We make no assumption that $p(a_t \mid h_t)$ is optimal. It is simply the conditional distribution induced by the language model's internal sampling mechanism.

The inferential objective under this model is the posterior distribution over trajectories:

$$p(a_{1:T} \mid o_{1:T}). \tag{2.2}$$

2.3. Sequential MAP Approximation

A common deterministic agent implementation selects actions greedily according to

$$a_t^{\text{greedy}} = \arg\max_{a \in \mathcal{A}} p(a \mid h_t). \tag{2.3}$$

This procedure induces a single trajectory

$$\hat{a}_{1:T}^{\text{greedy}}.$$

Observe that this is a sequential approximation to the global MAP trajectory

$$a_{1:T}^{\text{MAP}} = \arg\max_{a_{1:T} \in \mathcal{A}^T} p(a_{1:T} \mid o_{1:T}), \tag{2.4}$$

under the factorization (2.1).

However, the greedy rule (2.3) does not condition on future observations and collapses the trajectory distribution to a Dirac mass:

$$p_{\text{greedy}}(a_{1:T}) = \delta\Big(a_{1:T} - \hat{a}_{1:T}^{\text{greedy}}\Big).$$

No posterior mass is retained on alternative trajectories.

2.4. Probabilistic Forward Sampling

A probabilistic agent samples actions according to

$$a_t \sim p(a_t \mid h_t), \tag{2.5}$$

without maintaining weights over trajectories.

The resulting trajectory distribution equals the prior induced by the proposal model:

$$p_{\text{prior}}(a_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t).$$

This procedure explores multiple trajectories but does not incorporate the tool likelihood $p(o_t \mid a_t)$ into trajectory weighting.

2.5. Bayesian Sequential Monte Carlo Inference

To approximate the posterior (2.2), we introduce a particle system

$$\{a_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^{N},$$

where N is the number of particles.

Each particle evolves via proposal sampling

$$a_t^{(i)} \sim p(a_t \mid h_t^{(i)}),$$

where $h_t^{(i)} = (a_{<t}^{(i)}, o_{<t})$.

Weights are updated according to

$$w_t^{(i)} \propto w_{t-1}^{(i)}\, p(o_t \mid a_t^{(i)}), \tag{2.6}$$

with normalization

$$\sum_{i=1}^{N} w_t^{(i)} = 1.$$

Resampling may be applied to prevent weight degeneracy.
The empirical distribution

$$\hat{p}_N(a_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \, \delta\Big(a_{1:t} - a_{1:t}^{(i)}\Big)$$

provides a discrete approximation to the posterior $p(a_{1:t} \mid o_{1:t})$.

2.6. Inference Hierarchy

Under the unified model (2.1), we obtain the following hierarchy:

- Deterministic greedy inference corresponds to a single Dirac approximation of the posterior.
- Forward sampling corresponds to Monte Carlo sampling from the proposal prior.
- Sequential Monte Carlo corresponds to importance-weighted posterior approximation.

Thus, deterministic and probabilistic agents appear as special cases of inference under the same generative model, differing only in how uncertainty over trajectories is represented and propagated.

3. Structural Limitations of Greedy and Retry-Based Agents

We now analyze structural properties of deterministic and retry-based agent implementations under the unified model of Section 2. In particular, throughout this section we assume the finite action space $\mathcal{A}$, observation space $\mathcal{O}$, horizon T, and the joint factorization (2.1). Our goal is to make explicit which inferential approximations are implemented by common agent design patterns.

Throughout, we assume the finite action space $\mathcal{A}$, observation space $\mathcal{O}$, and generative model defined in Section 2.

3.1. Greedy Sequential Collapse

Recall that a deterministic greedy agent selects

$$a_t^{\text{greedy}} = \arg\max_{a \in \mathcal{A}} p(a \mid h_t), \tag{3.1}$$

thereby producing a single trajectory $\hat{a}_{1:T}^{\text{greedy}}$.

Under (2.1), the posterior $p(a_{1:T} \mid o_{1:T})$ generally has non-trivial support: unless the likelihood $p(o_t \mid a_t)$ and the realized observation sequence $o_{1:T}$ together force a unique feasible trajectory, there exist at least two distinct action sequences $a_{1:T} \neq a'_{1:T}$ with

$$p(a_{1:T} \mid o_{1:T}) > 0 \quad \text{and} \quad p(a'_{1:T} \mid o_{1:T}) > 0.$$

In contrast, greedy selection induces the trajectory distribution

$$p_{\text{greedy}}(a_{1:T}) = \delta\Big(a_{1:T} - \hat{a}_{1:T}^{\text{greedy}}\Big),$$

which collapses uncertainty to a singleton path and eliminates all alternative prefixes once a decision is made.

This collapse has a structural consequence: once a suboptimal action is selected at some time t, all alternative prefixes $a_{1:t}$ are eliminated from consideration. Since future actions depend only on the realized history h_{t+1}, no mechanism exists to reintroduce alternative branches. In Appendix A, we formalize this irrecoverability property and show that early mis-specification cannot be corrected under deterministic history-dependent policies.

From a filtering perspective, this corresponds to replacing the posterior distribution with a Dirac approximation at every time step. In classical state-space models, such deterministic collapse is known to be brittle in the presence of observation noise or model misspecification [1, 2].

3.2. Retry Mechanisms as Rejection Sampling

In practice, deterministic agents often employ retry loops when tool outputs fail validation (e.g., malformed JSON or constraint violations). Formally, suppose an agent samples

$$a_t \sim p(a_t \mid h_t),$$

executes the tool, and observes o_t. If o_t fails a binary validity test, the agent discards the result and resamples a_t.

Let $\mathcal{O}_{\text{valid}} \subseteq \mathcal{O}$ denote the subset of acceptable observations. The retry mechanism then implements sampling from

$$p(a_t \mid h_t, o_t \in \mathcal{O}_{\text{valid}}),$$

via rejection of proposals producing invalid observations.

This is a form of rejection sampling [3], where acceptance is determined by an indicator function

$$\mathbf{1}_{\{o_t \in \mathcal{O}_{\text{valid}}\}}.$$

Crucially, such retry schemes do not incorporate graded likelihood information $p(o_t \mid a_t)$ beyond binary validity. All accepted trajectories receive equal weight, regardless of how probable their observations are under the likelihood model. As a result, retry loops approximate conditional prior sampling rather than posterior inference.

In Appendix B, we formalize this observation and show that retry-based agents do not implement importance weighting and therefore cannot approximate $p(a_{1:T} \mid o_{1:T})$ except in degenerate cases.

3.3. Uncertainty Propagation and Support Preservation

Under the Bayesian Sequential Monte Carlo (SMC) formulation introduced in Section 2, the posterior is approximated by the weighted empirical measure

$$\hat{p}_N(a_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \, \delta\Big(a_{1:t} - a_{1:t}^{(i)}\Big),$$

with weights updated according to

$$w_t^{(i)} \propto w_{t-1}^{(i)} \, p(o_t \mid a_t^{(i)}).$$

Unlike greedy selection, this mechanism preserves multiple competing action prefixes so long as their likelihoods remain non-zero. Even if a particular particle proposes a suboptimal action at time t, alternative particles may retain support on other prefixes. As long as $p(o_t \mid a_t) > 0$ for multiple actions, posterior support need not collapse in a single step.

This behavior mirrors classical particle filtering in non-linear state-space models [1, 2]. While weight degeneracy may occur if the number of particles N is too small, collapse is no longer deterministic and can be controlled by resampling strategies.

In Appendix C, we provide a finite-space result showing that, under mild conditions, SMC preserves non-zero posterior support on multiple trajectories with non-zero probability, in contrast to deterministic greedy inference.

3.4. Relation to Sequential Decision Processes

The generative model (2.1) places tool-augmented agents within the broader class of partially observable sequential decision processes [4]. In such settings, maintaining a belief distribution over latent states is central to principled decision-making. The deterministic and retry-based agents described above correspond to extreme approximations of belief updates: either collapsing to a single state estimate or conditioning on binary validity events.

By contrast, the SMC formulation explicitly represents uncertainty over action trajectories, thereby aligning tool-augmented LLM agents with established probabilistic filtering methodologies.

The remainder of the paper builds on this unified view, and the appendix provides formal statements quantifying the structural distinctions between these inference regimes.

4. Bayesian Sequential Monte Carlo Agents

This section presents the Bayesian agent as an explicit posterior-inference procedure under the unified model (2.1). We adhere to the same notation as Sections 2–3: actions $a_t \in \mathcal{A}$, observations $o_t \in \mathcal{O}$, history $h_t = (a_{<t}, o_{<t})$, and horizon T.

4.1. Posterior Objective and Particle Approximation

Our inferential target is the trajectory posterior

$$p(a_{1:T} \mid o_{1:T}), \tag{4.1}$$

induced by the joint factorization

$$p(a_{1:T}, o_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t)\, p(o_t \mid a_t).$$

Sequential Monte Carlo (SMC) approximates the sequence of filtering distributions

$$p(a_{1:t} \mid o_{1:t}), \qquad t = 1, \ldots, T, \tag{4.2}$$

using a weighted empirical measure supported on N particles:

$$\hat{p}_N(a_{1:t} \mid o_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \, \delta\Big(a_{1:t} - a_{1:t}^{(i)}\Big), \tag{4.3}$$

This is a standard particle-filter construction [1, 2], specialized here to the latent variable $a_{1:t}$ (the action prefix) and observation stream $o_{1:t}$ (tool outputs).

4.2. Proposal and Weight Update

Given particles $\{a_{1:t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ approximating $p(a_{1:t-1} \mid o_{1:t-1})$, each particle is extended by sampling a new action from the LLM-induced proposal:

$$a_t^{(i)} \sim p(a_t \mid h_t^{(i)}), \qquad h_t^{(i)} = (a_{<t}^{(i)}, o_{<t}). \tag{4.4}$$

The tool is then executed (or simulated) to produce an observation $o_t \in \mathcal{O}$.[1]

Under the joint model (2.1) and proposal (4.4), the incremental importance weight is proportional to the likelihood term, yielding the update already introduced in Section 2:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \, p(o_t \mid a_t^{(i)}), \qquad w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}. \tag{4.5}$$

This update distinguishes Bayesian posterior reasoning from forward sampling: the likelihood $p(o_t \mid a_t)$ reweights trajectories based on how well proposed actions explain the observed tool outcomes.

4.3. Resampling and Degeneracy Control

A known issue in SMC is weight degeneracy: after repeated updates, a small number of particles may dominate. A common remedy is resampling when the effective sample size (ESS) falls below a threshold [1]:

$$\mathrm{ESS}_t = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}. \tag{4.6}$$

[1] In our experiments, o_t is the realized tool output. In analysis, we condition on the realized observation sequence $o_{1:T}$.

If $\mathrm{ESS}_t < \tau N$ for a chosen $\tau \in (0,1)$, we resample indices $I_1, \ldots, I_N$ i.i.d. from $\{1, \ldots, N\}$ with

$$\mathbb{P}(I_k = i) = w_t^{(i)},$$

set $a_{1:t}^{(k)} \leftarrow a_{1:t}^{(I_k)}$, and reset weights $w_t^{(k)} \leftarrow 1/N$. Other resampling schemes (systematic, stratified) may also be used; the specific choice does not affect our structural results in the appendix.

4.4. Decision Rule: From Posterior to Action

SMC produces a posterior approximation over action prefixes. To act, the agent must map $\hat{p}_N(a_{1:t} \mid o_{1:t})$ to either (i) a single selected action sequence, or (ii) a distribution over next actions. We use two common choices:

MAP Trajectory Estimate.. At any time t, define the particle-wise posterior mass on prefixes via weights. A natural estimate of the current best prefix is

$$\hat{a}_{1:t}^{\mathrm{MAP}} \in \arg\max_{i \in \{1,\ldots,N\}} w_t^{(i)}. \tag{4.7}$$

At horizon T, this yields a complete trajectory estimate $\hat{a}_{1:T}^{\mathrm{MAP}}$.

Posterior Predictive Action Selection.. Alternatively, select the next action according to the posterior predictive distribution induced by the weighted particles:

$$\hat{p}_N(a_{t+1} \mid o_{1:t}) = \sum_{i=1}^{N} w_t^{(i)}\, p(a_{t+1} \mid h_{t+1}^{(i)}), \quad h_{t+1}^{(i)} = (a_{1:t}^{(i)}, o_{1:t}). \tag{4.8}$$

This choice preserves uncertainty when committing to an action is premature (e.g., ambiguous intent).

Both rules reduce to deterministic greedy inference when $N = 1$ and the agent always chooses an arg max action under the proposal distribution.

4.5. Algorithm

Algorithm 1 summarizes the Bayesian SMC agent. The algorithm is stated in terms of the abstract model components $p(a_t \mid h_t)$ and $p(o_t \mid a_t)$, and therefore applies to any concrete implementation of the proposal (LLM prompting/sampling) and likelihood (tool noise model / parsing model).

Algorithm 1 Bayesian SMC Agent under (2.1)

Require: Horizon T, number of particles N, resampling threshold $\tau \in (0, 1)$, proposal $p(a_t \mid h_t)$, likelihood $p(o_t \mid a_t)$.

Initialize $a_{1:0}^{(i)} = \emptyset$ and $w_0^{(i)} = 1/N$ for $i = 1, \ldots, N$.
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    Sample action $a_t^{(i)} \sim p(a_t \mid h_t^{(i)})$, with $h_t^{(i)} = (a_{<t}^{(i)}, o_{<t})$.
    Extend prefix: $a_{1:t}^{(i)} \leftarrow (a_{1:t-1}^{(i)}, a_t^{(i)})$.
    Unnormalized weight: $\tilde{w}_t^{(i)} \leftarrow w_{t-1}^{(i)}\, p(o_t \mid a_t^{(i)})$.
  end for
  Normalize: $w_t^{(i)} \leftarrow \tilde{w}_t^{(i)} / \sum_{j=1}^{N} \tilde{w}_t^{(j)}$ for all i.
  Compute $\mathrm{ESS}_t = 1/\sum_{i=1}^{N} (w_t^{(i)})^2$.
  if $\mathrm{ESS}_t < \tau N$ **then**
    Resample N prefixes from $\{a_{1:t}^{(i)}\}$ with probabilities $\{w_t^{(i)}\}$; set all weights to $1/N$.
  end if
end for
Output: posterior approximation $\hat{p}_N(a_{1:T} \mid o_{1:T})$ as in (4.3); optionally output $\hat{a}_{1:T}^{\mathrm{MAP}}$ via (4.7).

4.6. Computational Complexity

At each step t, the agent proposes N actions and updates N weights. Ignoring the (implementation-dependent) costs of evaluating $p(a_t \mid h_t)$ and executing tools, the inference overhead is $O(N)$ per step, hence $O(NT)$ over a horizon T. Resampling can be implemented in $O(N)$ per resampling event.

In practice, the dominant cost is typically the proposal mechanism (LLM calls) and tool execution. The SMC formulation makes this cost explicit and controllable: increasing N improves posterior coverage but increases the number of proposals per step. This tradeoff is qualitatively explained in Section 7.

4.7. Discussion: Relation to Greedy and Forward-Sampling Agents

The SMC agent differs from forward sampling (2.5) by the likelihood-weighted update (4.5). It differs from greedy inference (2.3) by maintaining a distribution over competing prefixes rather than collapsing to a single path.

These distinctions are structural. In particular, the appendix proves that deterministic greedy inference eliminates alternative trajectories permanently, whereas SMC can retain non-zero posterior support on multiple prefixes (under mild finite-space conditions). This support-preservation property is the probabilistic mechanism enabling recovery from early mis-specification.

5. Likelihood Modeling for Tool-Augmented Agents

The Bayesian formulation in Sections 2–4 hinges on the likelihood term

$$p(o_t \mid a_t, h_t),$$

which quantifies how probable an observed tool response $o_t \in \mathcal{O}$ is under action $a_t \in \mathcal{A}$ given the interaction history $h_t = (a_{<t}, o_{<t})$.

We emphasize that the likelihood need not represent a perfectly calibrated probabilistic model of tool behavior. For Sequential Monte Carlo inference, it suffices that $p(o_t \mid a_t, h_t)$ encode relative plausibility of observations under competing actions.

Throughout, we maintain the finite spaces $\mathcal{A}$ and $\mathcal{O}$ defined in Section 2.

5.1. Structured Validation Likelihood

Many tool interactions produce structured outputs (e.g., JSON objects or schema-constrained responses). Let

$$V : \mathcal{A} \times \mathcal{O} \to \mathbb{R}_{\geq 0}$$

denote a validation loss measuring structural deviation of observation o_t under action a_t (e.g., parsing error, missing fields, type mismatch).

A natural likelihood construction is

$$p(o_t \mid a_t, h_t) \propto \exp\Big(- \lambda V(a_t, o_t)\Big), \qquad \lambda > 0. \tag{5.1}$$

The parameter λ controls the sharpness of structural enforcement. In the limit $\lambda \to \infty$, this reduces to a binary validity model:

$$p(o_t \mid a_t, h_t) \propto \mathbf{1}_{\{V(a_t, o_t)=0\}},$$

which corresponds to the retry mechanism analyzed in Section 3. Finite λ therefore strictly generalizes binary validation into graded posterior weighting.

5.2. Tool Reliability Likelihood

Tools may differ in reliability. Let $\epsilon_a \in [0, 1)$ denote the error rate associated with action $a \in \mathcal{A}$.

Let

$$C : \mathcal{A} \times \mathcal{O} \times \mathcal{H} \to \{0, 1\}$$

be a correctness indicator, where $\mathcal{H}$ is the history space and

$$C(a_t, o_t, h_t) = 1$$

indicates that the observation is judged correct for action a_t under context h_t (e.g., via verification, cross-checking, or consistency tests).

A simple reliability-aware likelihood is

$$p(o_t \mid a_t, h_t) \propto \begin{cases} 1 - \epsilon_{a_t}, & \text{if } C(a_t, o_t, h_t) = 1, \\ \epsilon_{a_t}, & \text{otherwise.} \end{cases} \tag{5.2}$$

This model allows posterior inference to prefer more reliable tools as evidence accumulates, without modifying the proposal distribution $p(a_t \mid h_t)$.

5.3. Semantic Consistency Likelihood

In many agent settings, correctness is semantic rather than structural. Let

$$S : \mathcal{A} \times \mathcal{O} \times \mathcal{H} \to \mathbb{R}$$

denote a semantic consistency score measuring agreement between o_t and the interaction history h_t under action a_t (e.g., embedding similarity or LLM-based scoring).

We may define

$$p(o_t \mid a_t, h_t) \propto \exp\Big(\gamma S(a_t, o_t, h_t)\Big), \qquad \gamma > 0. \tag{5.3}$$

This formulation penalizes inconsistent outputs without enforcing binary rejection.

5.4. Modular Combination

In practice, likelihood components may be combined multiplicatively:

$$p(o_t \mid a_t, h_t) \propto \exp\Big(-\lambda V(a_t, o_t)\Big) \cdot \Big[(1-\epsilon_{a_t})^{C(a_t,o_t,h_t)}\, \epsilon_{a_t}^{1-C(a_t,o_t,h_t)}\Big] \cdot \exp\Big(\gamma S(a_t, o_t, h_t)\Big). \tag{5.4}$$

Because SMC requires only likelihood ratios, normalization over $\mathcal{O}$ is unnecessary. Any non-negative function proportional to the desired plausibility weighting suffices for posterior approximation via (4.5).

This construction strictly generalizes retry-based agents and embeds them as a limiting special case within the Bayesian inference framework.

6. Implementation Architecture

We now describe how the unified probabilistic framework of Sections 2–5 maps to concrete tool-augmented LLM agent systems. The objective of this section is to make explicit the correspondence between model components and system-level modules, while preserving the inference hierarchy established earlier.

6.1. Mapping Model Components to System Modules

Under the generative model

$$p(a_{1:T}, o_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t)\, p(o_t \mid a_t, h_t),$$

each probabilistic component admits a direct implementation counterpart:

Model Component	**Implementation Module**
$p(a_t \mid h_t)$	LLM prompt + stochastic decoding
$p(o_t \mid a_t, h_t)$	Validation / scoring / reliability model
Particles $\{a_{1:t}^{(i)}\}$	Parallel agent branches
Weights $\{w_t^{(i)}\}$	Likelihood-based trajectory scores
Resampling	Branch pruning and duplication

The proposal distribution $p(a_t \mid h_t)$ is instantiated by prompting the LLM with the current history h_t and sampling from its decoding distribution (e.g., temperature-based sampling). No modification to the base language model is required.

The likelihood $p(o_t \mid a_t, h_t)$ is computed via structured validation, reliability scoring, semantic consistency, or combinations thereof as described in Section 5.

Particles correspond to independent agent branches, each maintaining its own action prefix $a_{1:t}^{(i)}$. Weights encode the plausibility of these prefixes given observed tool outputs.

6.2. Deterministic Agents as a Special Case

The classical deterministic tool-augmented agent corresponds to the degenerate configuration:

$$N = 1, \qquad a_t = \arg\max_{a \in \mathcal{A}} p(a \mid h_t),$$

with no likelihood weighting.

In this regime, the empirical posterior approximation reduces to a single Dirac mass:

$$\hat{p}_1(a_{1:t} \mid o_{1:t}) = \delta\Big(a_{1:t} - \hat{a}_{1:t}^{\text{greedy}}\Big).$$

Thus deterministic agents are recovered as a strict special case of the SMC framework with $N = 1$ and no resampling.

6.3. Probabilistic Forward Sampling as a Special Case

The probabilistic agent described in Section 2 corresponds to the configuration:

$$N > 1, \qquad w_t^{(i)} \equiv \frac{1}{N},$$

with no likelihood-based reweighting.

In this regime, particles explore multiple trajectories sampled from the proposal distribution $p(a_t \mid h_t)$, but no posterior correction is applied. The resulting empirical distribution approximates the prior over trajectories rather than the posterior.

Hence forward sampling appears as SMC without importance weighting.

6.4. Bayesian Agent with Posterior Weighting

The full Bayesian agent corresponds to the complete SMC algorithm of Section 4, including weight updates

$$w_t^{(i)} \propto w_{t-1}^{(i)}\, p(o_t \mid a_t^{(i)}, h_t^{(i)})$$

and optional resampling.

In implementation, this requires:

- Generating N candidate actions per step.
- Executing tools for each particle branch.
- Computing likelihood scores per branch.
- Maintaining and normalizing weights.
- Resampling when necessary.

No additional architectural components are required beyond parallel branch execution and scoring.

6.5. Computational Tradeoffs

Let C_{LLM} denote the cost of generating one action proposal and C_{tool} the cost of executing one tool call. At each time step, the SMC agent incurs cost

$$O\Big(N(C_{\text{LLM}} + C_{\text{tool}})\Big),$$

plus negligible overhead for weight updates and resampling.

The number of particles N therefore governs a direct tradeoff between:

- **Posterior coverage:** Higher N increases support preservation and robustness to early mis-specification.
- **Computational cost:** Higher N increases LLM and tool execution calls.

Deterministic agents correspond to the minimal-cost, maximal-collapse limit $N = 1$. Bayesian agents interpolate between these extremes.

6.6. Practical Considerations

In real systems, full parallel execution of all N branches may be impractical. Two implementation strategies mitigate this:

- **Asynchronous branch scheduling:** Evaluate only high-weight branches at each step.

- **Adaptive particle allocation:** Increase N when likelihood variance is high.

These strategies preserve the probabilistic interpretation while controlling computational overhead.

6.7. Summary

The probabilistic and Bayesian agents do not require fundamentally new system architectures. Rather, they reinterpret existing agent pipelines through an inference lens. Deterministic greedy selection, probabilistic sampling, and Bayesian SMC differ only in how uncertainty over action trajectories is represented and propagated.

This unified perspective enables principled reasoning about robustness, computational tradeoffs, and recovery from early decision errors, while remaining compatible with existing LLM and tool infrastructures.

7. Reference Implementation and Comparative Demonstration

We provide two independent reference implementations corresponding to the probabilistic and Bayesian regimes described in Sections 3–4:

- `probabilistic-langchain` — deterministic and forward-sampling agents with retry logic,
- `Bayesian-ProbabilisticLangchain` — particle-based Sequential Monte Carlo agent.

Both implementations share the same tool interface, action schema, observation structure, and history representation. The only architectural difference lies in the inference layer.

7.1. Shared Interaction Model

In both systems, interaction proceeds over a finite horizon T with:

- Actions a_t representing tool invocations,
- Observations o_t representing tool outputs or execution failures,
- History $h_t = (a_{1:t-1}, o_{1:t-1})$.

Tools expose a uniform `invoke(dict)` interface. Observations record success or failure status along with returned payloads. No tool-specific fallback logic is embedded in either framework.

7.2. Probabilistic and Retry-Based Runtime

The probabilistic framework implements two regimes:

Deterministic Greedy (DG).. A single tool is selected at each step. Upon failure, the same tool is retried up to a fixed budget R.

Probabilistic Forward Sampling (PF).. Actions are sampled from a proposal distribution (typically derived from the LLM), but only one sampled trajectory is executed per retry attempt.

In both cases, alternative hypotheses are discarded immediately after each step. The system never maintains multiple competing action prefixes simultaneously.

7.3. Bayesian Sequential Monte Carlo Runtime

The Bayesian framework maintains a population of N particles. At each step:

1. Each particle proposes candidate actions.
2. Each proposed action is executed, producing a particle-specific observation.
3. Weights are updated via

$$w_t^{(i)} \propto w_{t-1}^{(i)}\, p(o_t^{(i)} \mid a_t^{(i)}, h_t^{(i)}).$$

4. Particles are resampled when effective sample size falls below a threshold.

Particles therefore represent alternative action trajectories that coexist until evidence suppresses them through likelihood weighting.

7.4. Comparative Demonstration: Tool Reliability Under Fixed Budget

To make the inference distinction concrete, we consider a minimal two-tool environment:

- A flaky tool that fails with high probability (e.g., simulated timeouts).
- reliable tool that always succeeds.

The deterministic baseline selects the flaky tool and retries it up to five times. Under a fixed random seed, all five attempts fail. The Bayesian agent, operating under the same maximum step budget, proposes both tools within its particle set. Particles invoking the reliable tool receive high likelihood and dominate the posterior after weighting and resampling. The final selected trajectory therefore uses the reliable tool and succeeds.

Importantly, no tool-specific fallback logic is encoded. The reliable tool is not privileged; it simply accumulates posterior mass because its observed likelihood is higher. The difference in outcome arises entirely from inference structure:

- Retry-based runtime collapses onto a single action prefix.
- Particle-based runtime preserves multiple prefixes until evidence resolves them.

7.5. Interpretation

This demonstration illustrates the central thesis of the paper. Under identical tools and identical execution budget, deterministic retry fails while Bayesian inference succeeds. The improvement does not arise from model capacity or additional heuristics, but from maintaining posterior support over competing action trajectories.

The repository therefore serves as an executable instantiation of the formal model and appendix results: deterministic agents exhibit prefix collapse, whereas particle-based agents preserve alternative hypotheses long enough for evidence to accumulate.

8. Conclusion

Modern tool-augmented language model agents are typically engineered as deterministic or retry-based pipelines. In this paper we argue that many observed failures in such systems are not merely implementation artifacts, but consequences of implicit inference structure. When only a single action prefix is maintained at each step, errors at fragile intermediate stages can induce collapse from which recovery is unlikely.

We presented an alternative view: agent execution as sequential inference over action trajectories. Under this formulation, retry-based systems approximate posterior inference through repeated conditional sampling, while

particle-based methods preserve multiple competing hypotheses and resolve them through likelihood weighting. The distinction is structural rather than cosmetic.

We introduced a minimal Bayesian runtime that implements Sequential Monte Carlo over action sequences while preserving compatibility with existing tool interfaces. The probabilistic and Bayesian implementations share the same interaction contract and execution budget; they differ only in how uncertainty over trajectories is represented and propagated.

This work does not claim that particle-based inference universally dominates deterministic pipelines. Rather, it makes a narrower claim: when failures arise from premature commitment to an action prefix, preserving posterior support over alternatives provides a principled mechanism for recovery. The cost–robustness trade-off becomes explicit and tunable through particle count and likelihood design.

We view this as a reframing of agent engineering. Instead of adding increasingly complex fallback heuristics, one can modify the inference layer itself. Whether this perspective scales to larger multi-agent systems, adaptive likelihood models, or learned proposal distributions remains an open question.

Reference implementations:

- Probabilistic runtime (deterministic + sampling variants): https://github.com/AngshulMajumdar/probabilistic-langchain
- Bayesian Sequential Monte Carlo runtime: https://github.com/AngshulMajumdar/Bayesian-ProbabilisticLangchain

References

[1] A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carlo Methods in Practice, Springer, 2001.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking, IEEE Transactions on Signal Processing 50 (2) (2002) 174–188.

[3] C. P. Robert, G. Casella, Monte Carlo Statistical Methods, Springer, 2004.

[4] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1–2) (1998) 99–134.

Appendix A. Deterministic Policies and Irrecoverability

We work under the finite action space $\mathcal{A}$, finite observation space $\mathcal{O}$, horizon T, and generative model

$$p(a_{1:T}, o_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t)\, p(o_t \mid a_t, h_t), \qquad h_t = (a_{<t}, o_{<t}). \tag{A.1}$$

A deterministic history-dependent policy is a family of maps

$$\pi_t : \mathcal{H}_t \to \mathcal{A}, \qquad a_t = \pi_t(h_t),$$

where $\mathcal{H}_t$ denotes the finite history space at time t.

Lemma Appendix A.1 (Uniqueness of the action sequence given an observation sequence)**.** *Fix any observation sequence $o_{1:T} \in \mathcal{O}^T$. Under a deterministic policy $\pi = \{\pi_t\}_{t=1}^{T}$, there exists a unique action sequence $a^{\pi}_{1:T}(o_{1:T})$ generated by recursively applying $a_t = \pi_t(h_t)$ with $h_t = (a_{<t}, o_{<t})$.*

Proof. Since π_t is deterministic, $a_1 = \pi_1(\emptyset)$ is unique. Given $(a_{1:t-1}, o_{1:t-1})$, $a_t = \pi_t(a_{<t}, o_{<t})$ is uniquely determined. Proceeding recursively yields a unique $a^{\pi}_{1:T}(o_{1:T})$. □

Theorem Appendix A.1 (Irrecoverability of an alternative prefix within a fixed run)**.** *Let π be any deterministic history-dependent policy. Fix a time $t^\star \in \{1, \dots, T\}$ and a target action prefix $\bar{a}_{1:t^\star} \in \mathcal{A}^{t^\star}$. Consider any realized history $h_{t^\star+1} = (a_{1:t^\star}, o_{1:t^\star})$ generated by running π. If $a_{1:t^\star} \neq \bar{a}_{1:t^\star}$ at time $t^\star$, then, continuing the same run under the same deterministic policy π, it is impossible to later produce a complete action sequence whose first $t^\star$ actions equal $\bar{a}_{1:t^\star}$. Equivalently,*

$$\Pr(A_{1:t^\star} = \bar{a}_{1:t^\star} \mid h_{t^\star+1}) = 0.$$

Proof. Within a single run, the realized prefix $a_{1:t^\star}$ is already fixed at time $t^\star$. All future actions occur at times $t > t^\star$ and therefore cannot alter the first $t^\star$ actions. Hence no continuation can yield an action sequence whose prefix equals $\bar{a}_{1:t^\star}$ if it already differs at time $t^\star$. □

Remark Appendix A.1. Theorem Appendix A.1 is intentionally phrased as an in-run irrecoverability statement. It does not claim the policy cannot generate $\bar{a}_{1:t^\star}$ under different observation realizations in other runs; it states

that once the run has committed to a different prefix, deterministic policies cannot "reopen" that branch later.

Appendix B. Retry Mechanisms as Rejection Sampling

Fix a time t and history h_t. Consider an agent that repeatedly samples

$$a_t \sim p(a_t \mid h_t),$$

executes the tool to obtain $o_t \sim p(o_t \mid a_t, h_t)$, and accepts if $o_t \in \mathcal{O}_{\text{valid}}$ for a fixed validity set $\mathcal{O}_{\text{valid}} \subseteq \mathcal{O}$.

Theorem Appendix B.1 (Retry equals conditional prior sampling)**.** *The distribution of the accepted action A_t is*

$$\Pr(A_t = a \mid h_t,\ \textit{accepted}) = \frac{p(a \mid h_t)\,\alpha(a; h_t)}{\sum_{a' \in \mathcal{A}} p(a' \mid h_t)\,\alpha(a'; h_t)},$$

where the acceptance probability is

$$\alpha(a; h_t) = \sum_{o \in \mathcal{O}_{\text{valid}}} p(o \mid a, h_t).$$

Proof. Each proposal draw selects a with probability $p(a \mid h_t)$ and is accepted with probability $\alpha(a; h_t)$. Conditioning on acceptance yields the stated distribution by Bayes' rule / standard rejection sampling calculations [3]. Importantly, the accepted action depends only on the event $o \in \mathcal{O}_{\text{valid}}$ and not on graded likelihood of the realized observation. □

Appendix C. Support Preservation for Particle-Based Branching

This section formalizes the support-preservation phenomenon used by the Bayesian agent in the branching implementation described in Sectiosn 7 and 6 where multiple tool calls may be executed across particles.

At a fixed time t and history h_t, consider N particles. Each particle i proposes an action

$$a_t^{(i)} \sim p(a_t \mid h_t),$$

executes the tool to obtain a particle-specific observation

$$o_t^{(i)} \sim p(o_t \mid a_t^{(i)}, h_t),$$

and assigns an (unnormalized) weight

$$\tilde{w}_t^{(i)} = p(o_t^{(i)} \mid a_t^{(i)}, h_t), \qquad w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}.$$

(Resampling can be applied afterward but is not needed for the statement below.)

Theorem Appendix C.1 (Two-prefix support with non-zero probability)**.** *Assume there exist two distinct actions $a \neq a'$ in $\mathcal{A}$ such that*

$$p(a \mid h_t) > 0, \quad p(a' \mid h_t) > 0,$$

and there exist observations $o, o' \in \mathcal{O}$ with

$$p(o \mid a, h_t) > 0, \quad p(o' \mid a', h_t) > 0.$$

If $N \geq 2$, then with non-zero probability the normalized particle system after the weight update assigns positive mass to at least two distinct action choices at time t (equivalently, two distinct prefixes when appended to the same past prefix).

Proof. Because $p(a \mid h_t) > 0$ and $p(a' \mid h_t) > 0$, and $N \geq 2$, there is a strictly positive probability that at least one particle proposes a and at least one particle proposes a'. Conditional on those proposals, there is strictly positive probability that the corresponding tool draws produce o and o' respectively. Under the positivity assumptions, the unnormalized weights for those particles are strictly positive, so after normalization both receive strictly positive weight. Hence the empirical measure places positive mass on at least two distinct action extensions with non-zero probability. □

Remark Appendix C.1. Theorem Appendix C.1 is a finite-space support result aligned with the branching implementation. It does not claim convergence to the exact posterior $p(a_{1:t} \mid o_{1:t})$ (which would require conditioning on a shared observation stream). Its role is to formalize the non-deterministic collapse property: particle-based inference need not eliminate alternative branches in the deterministic way described in Appendix Appendix A.

Appendix D. Inference Ladder as Special Cases

We summarize the formal relationship among the three regimes under the unified model (A.1).

Proposition Appendix D.1 (Special-case relationships)**.** *Fix any realized observation sequence $o_{1:T}$. Under a deterministic policy π, the induced action sequence is uniquely $a^{\pi}_{1:T}(o_{1:T})$ from Lemma Appendix A.1. Thus deterministic greedy inference corresponds to a Dirac approximation conditional on $o_{1:T}$:*

$$\hat{p}(a_{1:T} \mid o_{1:T}) = \delta\Big(a_{1:T} - a^{\pi}_{1:T}(o_{1:T})\Big).$$

Probabilistic forward sampling generates trajectories from the proposal prior

$$p(a_{1:T}) = \prod_{t=1}^{T} p(a_t \mid h_t),$$

and particle-based branching augments proposals with likelihood-based weighting as in Theorem Appendix C.1.

Proof. The first statement follows directly from Lemma Appendix A.1. The forward sampling statement is immediate from the definition $a_t \sim p(a_t \mid h_t)$ without weighting. The final statement follows from the definition of particle proposals and weights. □

Appendix E. Reference Implementation

This appendix describes the reference implementation accompanying the paper. The objective of the repository is not to provide a production-grade agent framework, but to serve as a transparent and minimal realization of the inference ladder described in Sections 3–6.

Appendix E.1. Repository Structure

The repository is organized into three primary components:

- `p_langchain/` — probabilistic and retry-based agents (DG and PF).
- `b_langchain/` — Bayesian Sequential Monte Carlo agent.
- `demos/` — example scripts and experimental harnesses.

This separation enforces the central claim of the paper: deterministic, probabilistic, and Bayesian agents operate over the same tools, proposal mechanisms, and observation schemas, differing only in their inference layer.

Appendix E.2. Common Data Contract

Both implementations share an identical interaction contract:

Actions.. An action a_t encodes a tool invocation or terminal response and contains:

- a type (e.g., tool call or final answer),
- a tool name (if applicable),
- a structured payload.

Observations.. An observation o_t records:

- execution status (success, validation failure, partial),
- tool output or diagnostic message,
- auxiliary fields used to compute validation loss $V(a_t, o_t)$ or correctness indicators.

History.. The history $h_t = (a_{1:t-1}, o_{1:t-1})$ is identical across implementations. This ensures that performance differences arise strictly from inference and not from altered state representation.

Appendix E.3. Probabilistic and Retry-Based Agents

The `p_langchain` module contains:

- Deterministic Greedy (DG): single-action selection with retry upon validation failure.
- Probabilistic Forward Sampling (PF): action sampling from the proposal distribution with retry budget R.

Both agents implement what Appendix Appendix B formalizes as conditional prior sampling under binary acceptance. They do not maintain graded likelihood information across competing hypotheses.

Appendix E.4. Bayesian Sequential Monte Carlo Agent

The `b_langchain` module implements the particle-based agent described in Section 4. Each particle maintains:

- a partial trajectory $(a_{1:t}^{(i)}, o_{1:t}^{(i)})$,
- a weight $w_t^{(i)}$ updated via

$$w_t^{(i)} \propto w_{t-1}^{(i)}\, p(o_t^{(i)} \mid a_t^{(i)}, h_t^{(i)}).$$

Particles may execute different tools and therefore observe particle-specific outputs. Resampling is applied when the effective sample size falls below a threshold.

This branching execution model aligns with the support-preservation analysis in Appendix Appendix C.

Appendix E.5. Demos and Experimental Scripts

The `demos/` directory contains runnable scripts illustrating:

- metric-focused experiments (e.g., clock time, LLM calls, success rate),
- integration into standard retrieval-augmented generation (RAG) pipelines.

These scripts are provided to demonstrate reproducibility and integration feasibility. They are not required to understand the theoretical claims of the paper. The core distinction between frameworks lies in their inference mechanics rather than in task-specific engineering.

Appendix E.6. Design Philosophy

The implementation adheres to two principles:

1. **One-to-one functional mapping.** The tool set, proposal mechanism, and state representation are shared across probabilistic and Bayesian variants.
2. **Inference-only modification.** Performance differences arise solely from replacing retry-based collapse with posterior-weighted branching.

This architectural separation ensures that improvements attributed to the Bayesian agent stem from inference structure rather than from task-specific heuristics or tool-level modifications.