Soteria: Formally Verified Planning with Runtime Enforcement for Safe LLM Agents

Mike He[1] Ankush Desai[2] Sharad Malik[1] Aarti Gupta[1]

[1] PRINCETON UNIVERSITY [2] snowflake

① The Problem

An LLM agent that executes multi-step tool calls must meet two objectives at once:

Utility Complete the user's task.

Conformance Every tool call meets its contract and every policy rule holds over the trace.

They are in tension: blocking an unsafe call prevents the violation but can strand the agent, sometimes unrecoverably.

Runtime guardrails alone are not enough

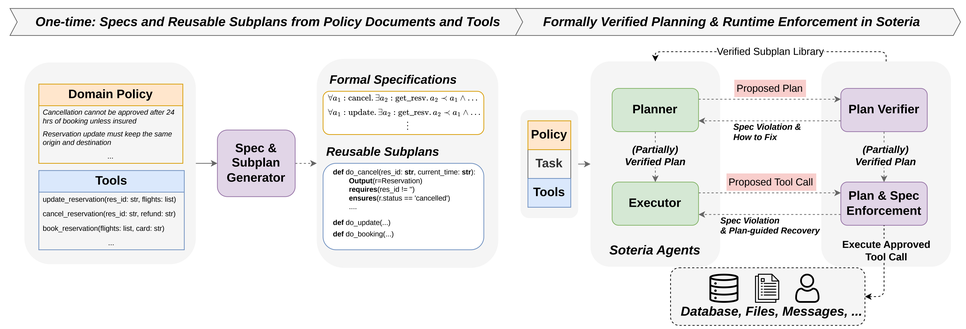
LLM judges a second model checks each action: easy to deploy, but there is *no formal guarantee*.

Runtime monitors contracts and trajectory invariants checked at runtime: provides guarantees at runtime, but cannot enforce liveness; no verified recovery path when a call is blocked.

Key insight

Agents can be safe without sacrificing utility by constructing a verified, specification-conformant plan before any tool is invoked; then using that plan at runtime as both an execution roadmap and a recovery guide.

② Soteria at a Glance



One-time setup (left): an auto-formalizer turns policy documents and tool definitions into specifications and reusable sub-plans, validated by hand. Per task (right): the Planner composes a hierarchical plan, the Plan Verifier verifies it agains policy specifications and returns repair diagnostics, and the Executor executes the plan under runtime enforcement of the specifications.

③ Specifications for Agent Correctness

Tool contracts a precondition on each tool's inputs and a postcondition on its outputs, constraining individual calls.

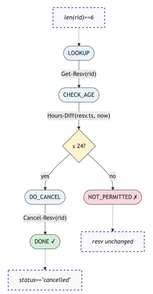
Policy correctness first-order formulas over the execution trace capturing temporal order (<), relational predicates and value flow (⇝) between calls.

Policy rule (τ^2-Airline)	Formalized specification
Check flown status before cancelling.	$\forall$ c:Cancel-Resv. $\exists$ c′:No-Flown. $c' < c \wedge c'.\text{rid} = c.\text{rid} \wedge c'.\text{r} = \text{true}$
Refund must derive from the reservation.	$\forall$ c:Refund-Resv. $\exists$ c′:Get-Resv. $c'.\text{r} \rightsquigarrow c.\text{amount} \wedge c.\text{rid} = c'.\text{rid}$

Upper: Any call to cancel reservation needs to be proceeded by a check that the flight has not been flown (< denotes happens-before);
Lower: The amount of refund must be derived from an actual reservation that is looked up earlier (⇝ denotes value-flows)

④ Plans as Guarded Transition Systems

A plan is a directed graph: states are stages of a procedure, transitions are tool calls, and every transition is gated by a guard: a Boolean condition over data collected at runtime. Plans are written in a Python-like DSL, making branching, sequencing and data dependence explicit and machine-checkable.



Hierarchical sub-plans

Each standard procedure is a pre-verified sub-plan with its own contract, invoked like a tool. The planner generates only the top-level composition, and verification is modular: a parent only reasons about a sub-plan's contract.

```python
# Plan: Cancel
def Cancel(uid :str, rid :str,
           reason :str, time :str):
  @state
  def LOOKUP(): # Initial state
    resv = call(Get-Resv, rid)
    goto(CHECK_AGE)

  @state
  def CHECK_AGE():
    hrs = call(Hours-Diff, resv.ts, time)
    goto(CHECK_RULES)

  @state
  def CHECK_RULES():
    if hrs <= 24:
      goto(DO_CANCEL)
    else:
      goto(NOT_PERMITTED)

  @state
  def DO_CANCEL():
    res = call(Cancel-Resv, rid, reason)
    goto(DONE)

  # Terminal states
  accepting(DONE)
  halting(NOT_PERMITTED)
```

Plan written in Soteria Plan DSL: State definitions, Actions, Guards and Plan Variables

⑤ Static Plan Verification

Soteria propagates a symbolic state Σ(s) over plan variables and discharges three families of verification conditions with Dafny:

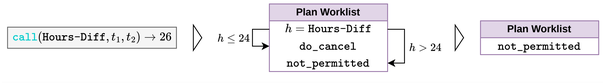
1. **Tool contracts** — $\Sigma(s) \wedge g \Rightarrow P_a$ at every call: no tool is invoked with invalid arguments.
2. **Progress** — Σ(s) is satisfiable at every state (no dead code) and $\Sigma(s) \Rightarrow g_1 \vee \ldots \vee g_n$ (no dead ends).
3. **Policy correctness** — a ghost state (step counter, call logs, provenance map) encodes <, relational and ⇝ predicates; assertions must hold on every accepting path.

Failures return targeted diagnostics that steer the planner to repair and re-verify within a budget of 8 retries; otherwise Soteria falls back to the strongest partially verified plan.

NEURAL INFORMATION PROCESSING SYSTEMS 2026, Atlanta

⑥ Runtime Enforcement

The plan is handed to the executor as a worklist: a flowchart progressively simplified as results arrive, collapsing the decided branch and showing only what remains.



SpecMonitor before every call, check the precondition, then evaluate every specification on the committed trace prefix extended with the proposed call. A call that would violate a spec is blocked before executing. Here, conformance holds whether or not the plan verified.

Recovery a blocked call sends the executor back to the worklist for an alternative branch instead of stranding it.

⑦ Results on 207 tasks · 3 benchmarks · 4 LLMs

100% conformance in all 12 model × benchmark settings

0 specification violations (baselines: 5–42)

1.7× the average success rate of the best baseline

τ^2-Airline (50)	GPT 5.4-mini	Gemini 3 Flash	Haiku 4.5	Opus 4.6
No Guardrail	34.0	28.0	30.0	52.0
NeMo	40.0	32.0	24.0	54.0
AgentSpec	50.0	36.0	26.0	56.0
ABC	48.0	26.0	22.0	50.0
Soteria	**60.0**	**62.0**	**70.0**	**88.0**

τ^2-Retail (114)	GPT 5.4-mini	Gemini 3 Flash	Haiku 4.5	Opus 4.6
No Guardrail	20.2	34.2	31.6	46.5
NeMo	18.4	28.1	23.7	38.6
AgentSpec	19.3	32.5	29.8	37.7
ABC	15.8	38.6	28.1	29.8
Soteria	**35.1**	**45.6**	**56.1**	**90.4**

Toolathon (43)	GPT 5.4-mini	Gemini 3 Flash	Haiku 4.5	Opus 4.6
No Guardrail	2.3	2.3	4.7	11.6
NeMo	0.0	0.0	0.0	9.3
AgentSpec	0.0	0.0	2.3	9.3
ABC	2.3	0.0	2.3	4.7
Soteria	**11.6**	**16.3**	**9.3**	**18.6**

Conformance-gated task success rate (%), Soteria in bold; best baseline conformance is 93.0% (ABC + Gemini 3 Flash, τ^2-Retail).

Ablation (τ^2-Airline, GPT-5.4-mini): flat unverified plan 44% success / 74% conformance; + verification 46 / 80; + sub-plans 50 / 80; + SpecMonitor 60 / 100. Sub-plans cut the median plan from 11 to 8 states and nearly double verified plans (65/68 vs 38/60).