Semantic Logic Structured Memory for Multi-Turn LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) exhibit severe unreliability in multi-turn underspecified language interactions, where critical information is revealed gradually across turns, leading to irreversible early commitments and compounding errors. Although many memory-based approaches have been thus studied recently, they implicitly treat dialogue history as flat token sequences or operate on opaque latent states. We argue that explicit semantic differentiation constitutes a critical representational layer to expose and revise early commitments. To this end, we propose Semantic Logic Structured Memory (SLSM), a language-grounded semantic state representation that explicitly tracks facts, unknowns, assumptions and constraints, which supports explicitly localized state revision across dialogue turns. Unlike slot-based dialogue state tracking or schema-driven memory, SLSM induces semantic structure dynamically from reasoning failures and constraint violations, rather than from predefined schemas. Evaluated on multi-turn instruction-following benchmarks, SLSM significantly improves reliability under sharded instruction settings, reducing outcome variance without sacrificing task performance. Our results suggest that explicit semantic state tracking is a critical component for robust multi-turn language reasoning under underspecification.

1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities in many real applications. However, in practice, multi-turn interactions are much more common. LLMs remain fundamentally unreliable in the multi-turn setting, as they are originally trained and deployed to maximize the performance in the single-turn manner, i.e., an objective mismatch.

One of the key challenges in multi-turn LLM interaction is under-specification, which refers to that each interaction only provides scattered underspecified dialogues, and critical information often incrementally arrives only after several turns. The current LLM behaviors tend to form premature commitments based on incomplete context, treating early responses as fixed anchors rather than provisional hypotheses. Once such early commitments are made, subsequent reasoning is frequently constrained or distorted by them, leading to compounding errors that are difficult to recover from as the dialogue unfolds. It thus requires the model to continuously track and revise its internal understanding of the task.

In response to under-specification, a growing body of work has been dedicated to developing memory mechanisms, aiming to manage information across dialogue turns. Existing approaches typically augment models with external memory buffers, long-context prompting, retrieval-augmented generation, or learned latent states that summarize prior interactions. While these techniques improve information persistence and mitigate context-length limitations, they largely treat dialogue history as undifferentiated text or compress it into opaque representations. As a result, the model's internal commitments remain implicit and entangled with surface tokens, making them difficult to identify, verify or revise when new information contradicts earlier assumptions.

Although memory mechanisms extend what LLMs can remember, they offer limited support for selectively revising what the model believes, especially under under-specified, evolving task semantics. In multi-turn interactions, effective reasoning requires distinguishing between (i) the information that is firmly established, (ii) the information that remains uncertain, and (iii) assumptions that are provisionally adopted to proceed with incomplete context. However, most existing memory designs do not make such distinctions explicit: facts, assumptions, intermediate inferences, and speculative hypotheses are stored or summarized in a uniform and flat representation. This lack of semantic differentiation prevents the model from performing localized revisions when new evidence arrives, forcing it instead to rely on global regeneration or implicit self-correction. As a consequence, early assumptions tend to persist beyond their validity, and contradictions are resolved implicitly through surface-level text generation, rather than principled state updates.

To address this representation-revision mismatch, we propose Semantic Logic Structured Memory (SLSM), a language-grounded semantic state representation system. Specifically, we design a structured state in each turn for semantic differentiation, consisting of fact, unknowns, assumptions, constraints and plan. It explicitly maintain and distinguish information that requires various actions in the multi-turn planning. To track and update the semantic state after each turn, we propose a semantic distillation module to extract the relevant elements from the semantic state. To selectively revise the semantic state for the next turn interaction, we propose an updater to incrementally revise the extracted relevant elements from the semantic distiller, and creating a next plan correspondingly.

We summarize our major contributions below:

- We develop a new LLM memory framework through rigorously defining the semantic state for the multi-turn LLM interaction. We also design a self-contained algorithm to explicitly tracking and selectively revise the semantic state in each turn.

- We perform extensive experiments to validate our argument that multi-turn LLM interaction suffers from unsatisfactory performance without tracking and selectively revising semantic states, and our proposed SLSM substantially improve the reliability of multi-turn LLM interaction.

2 Related Work

Multi-turn evaluation benchmarks. MultiChallenge (Deshpande et al., 2025)

LLM Gets Lost (Laban et al., 2025)

MT-Eval (Kwan et al., 2024)

AgentBoard (Chang et al., 2024)

LLM as judge

(Tang et al., 2025)

SOTA baselines for multi-turn LLM memory.

3 Proposed Method

In this section, we present Semantic Logic Structured Memory (SLSM) for multi-turn LLM interaction. We define a semantic state representation that explicitly maintain a structured semantic differentiation (Section 3.1), instead of encoding dialogue state implicitly in flat token sequences. Crucially, this semantic state is treated as an explicit first-class object, whose contents are deterministically updated and consulted at every turn,. We introduce an incremental update approach to maintain the semantic states over turns by selectively revising them as new information becomes available in Section 3.2.

3.1 Semantic State and Differentiation

We represent the multi-turn memory state at turn t as a structured tuple

$$S_t = (F_t, U_t, A_t, C_t, P_t),$$

where each component corresponds to a set of elements from distinct types of commitment with explicitly different epistemic status and revision rules, i.e., the fact set F_t, unknown set U_t, assumption set A_t, constraint set C_t and plan set P_t.

Facts. A fact $f \in \mathcal{F}$ is defined as

$$f = (\text{id}, \text{desc}, \text{src}, \text{turn}, \text{status}), \tag{1}$$

where id is an index for the fact, desc is a propositional description explicitly asserted by the user or returned by an external tool, src $\in \{\text{user}, \text{tool}\}$ denotes provenance, turn denotes the index of the turn deriving this fact, and status $\in \{\text{open}, \text{closed}\}$. Facts are treated as stable premises and are never introduced by model inference. The status field enables bookkeeping for scope expiration or task termination, without implying epistemic uncertainty.

Unknowns. An unknown $u \in \mathcal{U}$ represents a missing but necessary piece of information:

$$u = (\text{id}, \text{desc}, \text{required\_by}, \text{priority}, \text{status}), \tag{2}$$

where desc is a canonicalized description of the missing field, required_by specifies the dependent assumption or constraint, priority $\in \mathbb{R}^+$ indicates urgency, and status $\in \{\text{open}, \text{closed}\}$.

If the status of an unknown u is open, it indicates that task-critical information required for safe progression is still missing. As a result, the unknown induces a non-proceed action in the next turn by explicitly triggering a corresponding plan p, such as clarification or verification, which targets this unresolved element.

If the status is closed, the unknown has been resolved by newly acquired evidence, and its content is grounded in the semantic state. The resolved information is then instantiated as either an assumption a or a constraint c, depending on whether it represents provisional inferred content or an explicit requirement, respectively. Once closed, the unknown no longer participates in action selection.

Each unknown u must correspond to an askable question and be justified by an explicit dependency. Unlike predefined dialogue slots, unknowns in SLSM are dynamically induced by reasoning failure or constraint gaps, and do not assume a fixed schema.

Assumptions. An assumption $a \in \mathcal{A}$ is a provisional hypothesis introduced by the system:

$$a = (\text{id}, \text{desc}, \text{basis}, \text{confidence}, \text{turn}, \text{status}), \quad (3)$$

where basis $\subseteq \text{ID}(S)$ records supporting elements in a semantic state S by including their ids, confidence $\in [0, 1]$ is a self-assessed reliability score, and status $\in \{\text{valid}, \text{closed}, \text{contradicted}\}$.

Being valid means that the assumption is currently provisionally accepted as consistent with all known facts and active constraints, and is therefore permitted to support downstream reasoning and response generation. Importantly, validity does not imply truth; it only indicates that the assumption has not yet been challenged by contradictory evidence.

Being closed means that the assumption is no longer active in the current semantic state, not because it has been falsified, but because it has become irrelevant or superseded by state evolution. For example, when the task focus changes, the assumption's dependent unknowns are resolved in a way that renders the assumption unnecessary, or when the system transitions to a different plan that no longer relies on it. Closed assumptions are excluded from further reasoning without being treated as incorrect.

Being contradicted means that the assumption has been explicitly invalidated by new evidence, such as newly introduced facts or constraint violations that are incompatible with the assumption's content. Contradicted assumptions are actively marked as incorrect and may trigger localized revision, including the retraction of downstream inferences that depended on them.

Assumptions are explicitly retractable and never upgraded to facts. This prevents inferred hypotheses from being retroactively treated as ground truth, which is a key source of irreversible early commitment in multi-turn interaction.

Constraints. A constraint $c \in \mathcal{C}$ encodes explicit requirements:

$$c = (\text{id}, \text{desc}, \text{weight}, \text{turn}, \text{status}), \quad (4)$$

where desc is a declarative description, weight represents how strict this constraint is (higher means trying harder to satisfy it), and status $\in \{\text{satisfied}, \text{closed}, \text{violated}\}$.

Being satisfied means that the constraint is currently fulfilled by the existing facts and resolved unknowns in the semantic state, and therefore imposes no further restrictions on action selection or response generation.

Being violated means that the constraint is incompatible with the current semantic state. Specifically, no assignment of open unknowns can satisfy the constraint without retracting existing facts or valid assumptions. A violated constraint signals an explicit inconsistency and may trigger localized revision, verification, or corrective clarification in subsequent turns.

Being closed means that the constraint is no longer active in the current interaction, not because it has been violated or satisfied, but because it has become irrelevant due to state evolution. For example, when the task focus changes, when the constraint is superseded by a stronger requirement, or when the dialogue terminates. Closed constraints are excluded from further decision making without being treated as incorrect.

Plan. A plan $p \in \mathcal{P}$ specifies the system's operational mode and task for the next turn:

$$p = (\text{id}, \text{mode}, \text{target}), \quad (5)$$

where mode $\in \{\text{proceed}, \text{verify}\}$ is a specific action, and target $\in \text{ID}(S)$ denotes a target element in a semantic state S, on which the specific action will be on. The plan component does not encode long-horizon planning, but only the system's immediate epistemic stance toward the next turn. A plan with mode *proceed* marks the successful closure of the target element in memory, indicating that no further verification is required in subsequent turns.

Admissibility of Semantic States. Beyond defining the structure of semantic states, it is necessary to characterize when a state provides sufficient grounding for safely closing semantic elements. Admissibility captures whether the current semantic state permits assigning a *proceed* mode to a plan instance targeting a specific element, without risking premature commitment or inconsistency. Formally, a semantic state $S = (F, U, A, C, P)$ is said to be *closure-admissible* w.r.t. a target element if and only if all of the following conditions hold:

- All unknowns are resolved, i.e., there exists no $u \in U$ with status *open*;
- No assumption is contradicted by current evidence, i.e., there exists no $a \in A$ with status *contradicted*;
- All valid assumptions $a \in A$ satisfy a minimum *confidence* requirement τ;
- No constraint is violated, i.e., $c \in C$ with status *violated*.

If any of these conditions fails, the semantic state is non-admissible for closure, indicating that further clarification, verification, or revision is required before any plan instance may be marked as *proceed*. This admissibility notion is declarative and independent of any particular update or planning mechanism. In Section 3.2, we show how admissibility is checked after each semantic content update and how non-admissible states trigger selective revision and alternative epistemic modes.

Interaction-Level Semantics. These components collectively form an explicit semantic interaction layer that governs multi-turn behavior. **Facts** provide stable premises, **unknowns** surface missing information that blocks safe progression, **assumptions** enable provisional reasoning under uncertainty, **constraints** enforce admissible solution space, and **plans** mediate the system's epistemic stance toward the next turn. More importantly, these elements are jointly consulted to determine whether the system may proceed, must clarify, or needs to revise prior commitments. This ensures that interaction decisions are driven by explicit semantic state rather than implicit inference.

3.2 Incremental Update of Semantic States

Having defined the structure and epistemic roles of semantic states in Section 3.1, we now specify how such states evolve across turns. Multi-turn interaction is inherently a stateful process: each user utterance may introduce new evidence, resolve previously missing information, or invalidate provisional commitments. Therefore, a semantic memory is only meaningful if it supports principled and selective state revision over turns.

We define an *incremental, evidence-driven* update mechanism that (i) preserves the epistemic typing introduced in Section 3.1, (ii) avoids global rewriting of prior state, and (iii) makes all revisions explicit and auditable. Crucially, state updates must be justified by observable evidence rather than by implicit model preference or generation heuristics.

State Transition Interface. We model semantic state evolution as an incremental transition:

$$\Delta S_t \leftarrow \textsc{Update}(S_{t-1}, x_t), \tag{6}$$

where S_{t-1} denotes the semantic state from the previous turn, x_t is the current user input, and ΔS_t is a finite set of localized modifications. The updated state is obtained by applying ΔS_t to S_{t-1}, while all unaffected elements are preserved unchanged. This formulation enforces locality: revisions target specific semantic objects rather than re-encoding the entire dialogue history. It thereby enables selective correction of invalid commitments while stably preserving unrelated and already verified state elements.

Semantic Evidence Extraction. Before any admissibility checks and planning decisions are made, the current user input x_t is first interpreted as semantic evidence w.r.t. the existing state S_{t-1}. This step extracts candidate updates to the semantic components $(F_{t-1}, U_{t-1}, A_{t-1}, C_{t-1})$ without selecting a plan, which is formally written as follows:

$$(\Delta F_t, \Delta U_t, \Delta A_t, \Delta C_t) \leftarrow \textsc{Ext}(S_{t-1}, x_t), \tag{7}$$

where Ext maps the raw utterance into typed semantic evidence, and explicitly isolates the change. Importantly, this extraction step is non-committal: it may introduce, resolve or contradict semantic elements, but it does not determine whether the system should proceed or verify.

The extraction follow the following principles. First, only propositions explicitly asserted by the user or returned by external tools may enter the fact set. Second, missing but task-relevant information is surfaced as unknowns rather than silently instantiated. Third, inferred or defaulted content is introduced, when necessary, as assumptions with explicit confidence. Fourth, explicit requirements expressed in the input are recorded as constraints. These operations update the semantic content of the state while leaving the plan component undecided.

We denote the resulting intermediate state as S_t^-, which reflects all semantic information grounded in x_t but has not yet undergone admissibility checking or action selection.

Evidence-Driven State Revision. When any of $\Delta F_t, \Delta U_t, \Delta A_t$ and ΔC_t is non-empty, updates are applied in a type-aware and localized manner:

- **Facts** may only be added or marked as closed based on explicit user input or tool output; they are never introduced or altered by the system inference.

- **Unknowns** are marked as *open* when task-relevant information is missing. Open unknowns remain explicitly represented and are never silently instantiated. When missing information is revealed through user interaction or external tools, the corresponding unknown is resolved and converted into an assumption with *valid* status and a system-evaluated confidence. The introduction of this new assumption does not by itself guarantee admissibility, so its consistency with existing assumptions and constraints is evaluated in the admissibility check at the state level.

- **Assumptions** may be marked as *valid*, *closed* or *contradicted*, depending on newly available evidence. Importantly, assumptions are retractable and are never promoted to facts.

- **Constraints** are evaluated against the current state and marked as *satisfied*, *violated*, or *closed* without being implicitly discarded.

Revisions affect only the minimal set of state elements, whose status is justified by new evidence; unrelated elements are guaranteed to remain unchanged. This non-interference property prevents cascading side effects and supports fine-grained correction of early commitments.

Admissibility Check. After semantic evidence extraction and localized state revision, the system must determine whether the updated semantic state provides sufficient grounding to safely close semantic elements. In SLSM, this decision is formalized through an admissibility check, which evaluates whether a plan instance targeting a specific element may be assigned the *proceed* mode without introducing premature commitment or unresolved inconsistency. Concretely, a semantic state is considered non-admissible for closure whenever it violates any of the following epistemic preconditions:

1. **Unresolved unknowns.** There exists an unknown $u \in U_t$ with status *open*. Since open unknowns represent task-critical missing information, proceeding would require silent value imputation, which is disallowed.

2. **Contradicted assumptions.** There exists an assumption $a \in A_t$ with status *contradicted*. Such assumptions explicitly encode inconsistency with current evidence and must be revised before further reasoning.

3. **Excessive assumption uncertainty.** There exists a valid assumption whose confidence falls below a predefined threshold, indicating that the current state relies on unstable speculative commitments.

4. **Violated constraints.** There exists a constraint $c \in C_t$ with status *violated*, implying that the current state is incompatible with explicit task requirements.

Whenever any of these conditions is met, the current semantic state is not closure-admissible. In such cases, no plan instance can be assigned the *proceed* status in the next turn, and the system must instead favor epistemic memory operations, i.e., verification to revise the semantic state.

Admissibility gating prevents premature epistemic closure in multi-turn interaction. In standard LLM dialogue, speculative choices made under missing information are immediately committed in surface text and become difficult to retract, leading to compounding errors. By decoupling interaction continuity from memory closure, SLSM preserves uncertainty as explicit unknowns and retractable assumptions, which ensures that early speculative decisions do not prematurely solidify or constrain subsequent state revision.

Coupling State Update with Plan Derivation. Semantic state updates directly determine the system's epistemic stance for the next turn by inducing the set of admissible memory-management plans. Given the updated semantic state components F_t, U_t, A_t, and C_t, the plan set P_t is derived by selecting plan instances whose modes are compatible with the current admissibility status of the state. In particular, unresolved task-critical unknowns and violated constraints induce verification plans, whereas a plan instance (_, proceed, target) is admitted only if the current semantic state is admissible with respect to the target element target. Thus, plan derivation is not governed by ad hoc dialogue policies, but is systematically induced by semantic state properties.

With explicit state tracking and selective revision, state-induced action selection completes the SLSM control loop: semantic evidence updates the state, the state induces appropriate actions, and actions determine whether the system seeks information, verifies beliefs, or proceeds to solve the task. This closed-loop design is critical for maintaining stability and recoverability in underspecified multi-turn interactions.

4 Experiments

4.1 Objectives

Objective 1: Necessity for Preventing Premature Commitment. Hypothesis (O1). In underspecified multi-turn interactions, *preventing premature commitment and avoiding lock-in of early speculative decisions* requires explicit semantic differentiation with localized revision; memory capacity, retrieval, or summarization alone are insufficient. This robustness can be achieved without degrading final task accuracy.

Objective 2: Causal Role of Explicit Revision. Hypothesis (O2). The reduction of premature commitment in SLSM is causally enabled by explicit tracking and localized revision of assumptions. Disabling revision negates the robustness benefits of semantic differentiation, even when all other components are preserved.

Objective 3: Structural Necessity of Semantic Components. Hypothesis (O3). The semantic components in SLSM (facts, unknowns, assumptions, constraints) play complementary roles in preventing premature commitment and lock-in. While a full factorial ablation is beyond the scope of this submission, we expect that removing a key component (e.g., assumptions) will measurably degrade robustness under identical model capacity and token budgets; we provide a targeted ablation to partially support this hypothesis.

Objective 4: Failure-Mode Specificity. Hypothesis (O4). SLSM mitigates a distinct class of structural failure modes—specifically premature assumption commitment and irreversible lock-in—that persist across state-of-the-art memory and agent-based baselines, rather than merely improving average task accuracy.

Experimental Task Checklist (Frozen)

Global Setup (Day 0)

- Implement unified API-only runner (prompt $\rightarrow$ response $\rightarrow$ JSONL)
- Add caching and retry logic
- Fix model and decoding (temperature = 0)
- Define baselines: Plain Chat, Naive Prompt Memory
- Fix unified output schema for all benchmarks

MultiChallenge (Day 1)

- Integrate official dataset and evaluator
- Generate final-turn responses (Plain / Prompt / SLSM)
- Run official metrics
- Export main table (overall + category-level scores)
- Log failure and error cases

LLMs Get Lost (Day 2)

- Integrate session/trajectory format
- Generate per-turn responses (minimal required)
- Compute lostness and task completion metrics
- Export main table
- Extract 3–5 representative cases

MT-Eval (Day 3a)

- Integrate official tasks and evaluator
- Generate final responses
- Run official scoring
- Export summary table (main or appendix)

AgentBoard (Day 3b)

- Select dialogue/memory-only subset
- Generate responses
- Run corresponding metrics
- Export appendix table

Figures and Paper Integration (Day 3c)

- Consolidate all tables (uniform formatting)
- Generate one normalized bar plot across benchmarks
- Write experimental setup paragraph (model, decoding, metrics)
- Write benchmark scope and subset declaration

Explicitly Excluded

- LLM-as-judge experiments
- Multiple model sweeps
- Multi-seed or stochastic decoding
- Full-track AgentBoard evaluation

References

Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Yuqi Tang, Kehua Feng, Yunfeng Wang, Zhiwen Chen, Chengfei Lv, Gang Yu, Qiang Zhang, and Keyan Ding. 2025. Learning an efficient multi-turn dialogue evaluator from multiple judges. *arXiv preprint arXiv:2508.00454*.

A Example Appendix

This is an appendix.

B Related Work

Text-based / retrieval-based memory

Latent / learned memory

Structured / schema-based memory

Dialogue State Tracking

Belief State / Belief Tracking(POMDP / Dialogue Policy)

Schema-based / JSON / Table Prompting

Knowledge Graph / Dynamic KG Updating

Non-monotonic Reasoning / Belief Revision