Commitment-Carrying Agent State

A Missing Primitive for Persistent AI Agency

Chaitanya Mishra

Independent Researcher

April 2026

Abstract

AI agents remain brittle over long horizons not chiefly because they lack more memory, more planning depth, or more tools, but because they lack a stable decision state. Existing systems replay transcripts, retrieve memories, or maintain ad hoc summaries, yet they routinely collapse observations, assumptions, obligations, hypotheses, plans, and delegated promises into undifferentiated text. I call this failure *semantic state collapse*. It causes dormant constraints to vanish until the instant they matter, self-authored hypotheses to return later as evidence, parent objectives to be lost during delegation, and stale plans to survive world changes.

This paper argues that persistent agency requires a missing systems primitive: *commitment-carrying agent state* (CCAS). In CCAS, every future-relevant semantic item is represented as a typed commitment with support provenance, validity guards, discharge conditions, ownership, priority, and refinement links. The agent becomes a state transformer over this object rather than a policy that repeatedly reconstructs intent from raw dialogue history. I formalize open-world agent execution as reasoning over event histories containing external evidence, self-authored artifacts, actions, and messages; define support-sensitive task families on which untyped state must fail; and prove two core results: first, action invariance under commitment-sufficient compression, and second, delegation safety under contract refinement.

The paper also introduces a concrete runtime architecture, a dual-language state schema, an action authorization boundary, an epistemic firewall preventing self-generated text from silently becoming evidence, and a *decision-state bottleneck* objective for learning compact but commitment-sufficient state abstractions. Because no experiments are run here, the empirical contribution is a rigorous evaluation program: benchmark families for dormant constraint activation, provenance-sensitive reasoning, revalidation under world drift, and contract-preserving delegation, together with metrics and ablations targeted at the theory's failure signatures.

The central thesis is simple. Long-horizon agency is not just a memory problem. It is a state problem. A transcript is a log, not a control state. If AI agents are to become reliable autonomous systems rather than eloquent stateless emulations, they need a semantics-preserving substrate for what they are committed to, why, and under what conditions those commitments remain live.

Contents

1 Introduction

Autonomous language-model systems now plan over repositories, browse the web, operate graphical interfaces, write code, consult tools, and coordinate with other agents across trajectories that can last hundreds of steps or more [1, 2, 4, 7, 9–11]. Their short-horizon competence can be striking. Their long-horizon behavior is far less stable. An engineering agent respects a repository policy for thirty minutes and then violates it at the moment of merge. A research agent invents a hypothesis early in an inquiry and later cites its own note as if it were evidence. A procurement agent hands off vendor review to a subagent and receives a locally reasonable recommendation that quietly breaks a global legal or budgetary constraint. A travel assistant carries a "book only after visa approval" instruction across many turns, until a late-stage price drop induces it to act before the prerequisite is satisfied. These failures feel different at the surface. Structurally they are the same.

The common pathology is that current agents usually persist *logs*: transcripts, summaries, retrieved snippets, reflections, plans, code artifacts, or vector-store memories. They do not persist a clean, typed, semantically grounded *decision state*. ReAct-style loops interleave thought and action effectively, but the state that matters is still hidden in text [1]. Tree-of-Thoughts improves local search but the thoughts remain untyped candidates rather than durable commitments [3]. Reflexion stores self-authored feedback in an episodic buffer, which is useful for learning from failure, but the resulting memory still mixes externally grounded observations with self-generated interpretations [4]. MemGPT and related systems manage memory hierarchies more intelligently, yet memory-tier management does not by itself define what the agent is normatively or epistemically committed to [8]. Even recent proposals for persistent or bounded internal state are closer to the right abstraction while still leaving crucial semantics implicit [26–28].

This paper advances a stronger claim: the missing primitive is not "more memory" but *commitment-carrying state*. A capable long-running agent needs an explicit, machine-queryable object that stores what must remain true or be treated as tentatively true, which of those items are hard constraints or soft preferences, what evidence supports them, when they become applicable, when they are discharged, which later plans depend on them, and how they are exported to subagents. Without such an object, every future action requires re-deriving semantic commitments from unstructured history. That reconstruction may be possible for short episodes, but it is not a scalable architecture for persistent agency.

I call the deep failure mode *semantic state collapse*. In semantic state collapse, an architecture preserves enough surface information to continue a conversation or task, but it fails to preserve the semantic distinctions that make future control reliable: observation versus assumption, obligation versus preference, hypothesis versus evidence, parent-level constraint versus child-level subgoal, live commitment versus stale residue. The architecture may still look coherent in natural language. It is no longer acting from the same state.

The central proposal is *commitment-carrying agent state* (CCAS), a systems primitive analogous in spirit to a process control block in operating systems, except that the relevant resources are

semantic. A process log is not enough to resume a program safely; one also needs registers, memory maps, privileges, open descriptors, and scheduling state. Likewise, an event log is not enough to resume an agent faithfully. One needs a compact control object that carries the agent's active commitments and the support structure behind them. In CCAS, persistent state is a typed graph whose nodes are evidence atoms and commitments; edges encode support, conflict, refinement, dependence, and discharge; and runtime components compile history into state, gate actions through hard commitments, and propagate contracts during delegation.

The contribution is not that commitments, provenance, or agent contracts are individually new. Classical agent research long emphasized beliefs, goals, and intentions [20, 21]; multiagent systems developed sophisticated commitment-based interaction semantics [22–24]; truth-maintenance and provenance systems studied support tracking in symbolic settings [17, 18]; and proof-carrying code showed how explicit certificates can make untrusted execution safer [19]. What is missing in contemporary AI agents is the *joint unification* of these ideas into the agent's persistent control state. The paper's thesis is therefore architectural, not merely taxonomic.

The paper makes six main contributions.

1. It identifies *semantic state collapse* as a fundamental failure mode underlying goal drift, provenance loss, stale planning, and delegation failure in long-horizon agents.
2. It formalizes open-world agent execution and defines *commitments*, *support-sensitive tasks*, *semantic state collapse*, and *commitment-sufficient abstraction*.
3. It proves an impossibility-style result for untyped persistent state on support-sensitive task families, and proves safety and resumability properties for commitment-sufficient state and refinement-preserving delegation.
4. It proposes CCAS, a concrete state substrate with typed commitments, provenance links, guards, discharge rules, action authorization, and delegation contracts.
5. It introduces a *decision-state bottleneck* objective for learning compact state abstractions that preserve action-relevant commitments rather than merely compressing history.
6. It defines an evaluation program, including benchmark families, metrics, baselines, and ablations tailored to the theory's distinctive empirical signatures.

The rest of the paper proceeds from diagnosis to formalization to mechanism. Section 2 compares several candidate deep problems and explains why semantic state collapse subsumes the strongest alternatives. Section 3 argues that logs are not state and extracts design requirements for persistent agency. Section 4 shows why current agent paradigms, including recent memory-control proposals, remain incomplete. Section 5 gives the formal problem statement. Section 6 and Section 7 introduce the proposed state substrate and training objective. Section 8 presents theoretical results. Sections 9 to 11 develop systems implications, evaluation, and worked cases. Later sections pressure-test the proposal, discuss limitations, and situate it within related work.

2 Competing Diagnoses and Research Thesis

Before proposing a new primitive, it is necessary to ask whether a different diagnosis would better explain the recurrent failures of agent systems. Several candidate "fundamental problems" are plausible. Some are undeniably important. The question is which one is deepest, most general, and most actionable.

2.1 Candidate diagnoses

Planning and search deficiencies. A natural diagnosis is that agents fail because they do not search enough. This view is supported by the gains from deliberative prompting and tree search [3]. It explains myopic action choice, brittle decomposition, and local reasoning errors. It does not explain why a system can search competently while still violating an instruction that was stated clearly earlier and remains globally binding. Search chooses among available branches. It does not by itself preserve the semantic objects that define which branches are admissible.

Tool unreliability and poor environment interfaces. Another diagnosis is that agents fail because tools are noisy or because interfaces are not designed for machine use. This is partly true. Better action interfaces can materially improve performance [11]. Yet many failures of long-horizon autonomy occur even when the tools are perfect. A subagent can return exactly the data requested while still violating a global parent constraint. A model can quote the correct document while still misclassifying a hypothesis as evidence. Interface quality matters, but it does not resolve how semantic commitments persist across time.

Memory shortage. Long context is clearly relevant. Retrieval-augmented systems and memory hierarchies help agents recover information omitted from the active context [8, 12]. Recent work rightly argues that persistent or bounded state is preferable to endless transcript replay [26–28]. But memory as storage is still not the same as state for control. A vector store can retrieve the sentence "do not change the public API," yet still fail to represent it as an active hard constraint with a watch-set over changed signatures. The problem is not merely whether the information is stored. It is whether its role in future control is explicit.

Goal drift. Goal drift is a serious long-horizon phenomenon and has recently begun to receive direct empirical study [25]. It captures how agents deviate from their initial objective under contextual pressure. However, drift is best understood as one major symptom of a deeper representational problem. An agent can preserve its high-level goal and still fail because it forgets a specific obligation, mistakes tentative reasoning for evidence, or passes an underspecified contract to a subagent. Goal drift addresses objective continuity. It does not fully address epistemic continuity or delegation fidelity.

Table 1: Competing diagnoses of long-horizon agent failure. The table is intentionally comparative rather than rhetorical: each diagnosis identifies something real, but only the last offers a single abstraction that naturally subsumes the others.

Diagnosis	Explains well	Misses or under-explains	Verdict
Planning/search insufficiency	Myopic errors, poor decomposition, local dead ends	Why early obligations disappear, why hypotheses later masquerade as evidence, why delegation loses parent intent	Important but downstream of state
Tool/interface unreliability	Perception failures, malformed actions, weak environment affordances	Failures with perfect tools or clean human handoff artifacts	Necessary but not fundamental
Memory shortage	Context overflow, missing recall, long-range reference failure	Normative status, provenance, guards, discharge, refinement, resumability	Too storage-centric
Goal drift	Objective deviation under pressure, slow behavioral shift	Provenance failures, stale plans, subagent contract leakage without explicit goal change	A major symptom
Multi-agent coordination inefficiency	Message overload, specialization failure, role conflict	Single-agent long-horizon failures and self-handoff problems	Secondary effect
Semantic state collapse	Dormant constraint loss, provenance confusion, stale planning, delegation failure, goal drift under context pressure	Does not remove the need for good planning, tools, or memory mechanisms	Best unifying thesis

Multi-agent communication and coordination failures. Large parts of the field now focus on multi-agent workflows, role assignment, and communication protocols [5]. Those issues are real, but the same failure patterns appear in single-agent systems that simply span many steps or interact with future versions of themselves through summaries, reflections, or checkpoints. The root difficulty is not communication volume. It is the absence of a trustworthy state interface.

2.2 Why semantic state collapse is the strongest thesis

Semantic state collapse dominates the alternatives on six dimensions: generality, novelty, depth, tractability, leverage, and explanatory power.

The decisive point is that semantic state collapse explains why superficially different fixes repeatedly help a little and then plateau. Better planning helps because it temporarily reconstructs

missing state. Better memory helps because it improves the odds that the relevant commitment can be rediscovered. Better tools help because cleaner observations make the reconstruction task easier. Better evaluation helps because some benchmarks finally expose the failure. Yet none of those fixes directly installs the missing primitive itself.

The chosen thesis is therefore:

> *Persistent AI agency is limited by the absence of a semantics-preserving decision state. Current agents preserve history, memories, and artifacts, but not a first-class state of commitments. As a result they are unable to reliably maintain semantic continuity across long-horizon reasoning, action, delegation, and world change.*

This thesis is also tractable. Unlike a vague complaint that "agents should reason better," it suggests precise definitions, algorithmic interfaces, theoretical properties, and falsifiable benchmarks. One can test whether a state representation preserves dormant constraints, whether self-authored hypotheses can contaminate evidence channels, whether delegation contracts preserve parent commitments, and whether checkpointed agents resume with the same admissible action set.

3 Logs Are Not State: Background and Design Requirements

3.1 From histories to control state

Sequential decision making has long distinguished between a full history and a sufficient state abstraction. In a partially observable process, an agent cannot act well from the latest observation alone; it needs a summary of history sufficient for future control [13]. Predictive state representations and later work on state abstraction refine this idea further [14, 15]. Classical agent architectures also separated beliefs, desires, and intentions to avoid re-inferring everything from scratch every cycle [20, 21]. Contemporary language-model agents often collapse back to the opposite extreme: the transcript or an approximate summary becomes the de facto state.

That collapse is understandable. Large language models are strong sequence models. It is natural to let the model read history and reconstruct whatever matters on demand. For short tasks this is often enough. The trouble is architectural rather than informational. A raw history is a *log*. A control state is a compact object that supports continuity under bounded recomputation. Systems engineering has learned this lesson repeatedly. Event logs are valuable for replay and debugging, but they do not replace process control blocks, database indexes, or transactional snapshots. Those additional objects exist because efficient, safe continuation requires more than archival fidelity.

The same distinction matters for agents. A log of everything the agent has seen and said can be perfect and still fail to provide persistent agency. Continuation would require the model to re-derive, from scratch at every major step, which instructions are still binding, which hypotheses remain tentative, which obligations are dormant but armed by future guards, which delegated subtask

outputs may update the parent state, and which old plans should be invalidated by new evidence. The more the horizon grows, the more this repeated re-derivation becomes fragile.

3.2 Why open-world agents need more than belief state

One might object that belief state already solves the problem. In a classical POMDP, a belief over latent environment state is sufficient for control. Why not let the agent's memory or latent activations play that role? The answer is that open-world agents operate over a richer semantics than environment uncertainty alone.

First, relevant future control depends not just on facts about the world, but on *normative and procedural commitments*. "Do not merge code without preserving the public API." "Do not spend above the approved budget." "Only cite claims supported by external sources." These are not ordinary latent world variables. They are filters on admissible action.

Second, open-world agent histories contain *self-authored artifacts*: plans, reflections, summaries, hypotheses, issue triage notes, and partial conclusions. Such artifacts may be useful, but they should not automatically inherit the same epistemic status as externally grounded evidence. A purely belief-state view has no natural place to represent that distinction.

Third, long-running agents frequently *delegate*. Parent and child agents can have different scopes, permissions, and success conditions. The relation between parent objectives and child objectives is one of refinement, not simple state transition. Existing multi-agent frameworks usually encode this relation in prompts, conventions, or code paths rather than in a first-class contract object.

Fourth, the world itself can change while commitments remain live. A plan that was justified yesterday may be invalidated today by a tool output, a message, or a newly observed precondition failure. Maintaining coherence in this setting requires explicit revalidation machinery.

For these reasons, belief state is part of the story but not the whole of it. A modern agent needs a control state that combines epistemic, normative, and delegation semantics.

3.3 Design requirements for persistent agency

The missing primitive should satisfy at least five requirements.

R1. Commitment continuity. If an instruction, obligation, reservation, or chosen objective remains live, it must survive long irrelevant stretches and reappear when its guard becomes active.

R2. Epistemic integrity. The system must distinguish observations, derived beliefs, hypotheses, and self-authored notes. It should be able to reconstruct why a claim is held and what kind of support licenses it.

R3. Delegation fidelity. When a parent agent delegates, the child's local objective must be a refinement of the parent's relevant commitments, not an informal restatement that can silently omit constraints.

R4. Revalidation under drift. When the world changes or new evidence arrives, the system must know which plans, assumptions, and downstream commitments depend on that support and should therefore be revised, suspended, or discharged.

R5. Resumable agency. An agent that is reset, resumed on another machine, or handed from one executor to another should continue from a persistent semantic state, not merely from a textual summary whose control implications must be re-inferred.

Requirements R1 through R5 jointly imply that the persistent object must be more than a memory store. It must be a typed, queryable, updateable state with explicit support and validity structure.

4 Why Current Agent Paradigms Fail at This Problem

This section is deliberately unsentimental. Contemporary agent work has achieved real progress. Yet across paradigms, the same missing primitive keeps reappearing.

4.1 Transcript-conditioned agent loops

ReAct-like scaffolds interleave natural-language reasoning and action selection in a single loop [1]. Their strength is flexibility. Their weakness is that their persistent substrate is still mostly a prompt transcript, perhaps with truncation, summarization, or retrieval. Long-horizon semantic continuity then depends on the model repeatedly rediscovering which pieces of the transcript are still relevant and what role each piece plays. An early instruction, a later caveat, and the agent's own plan all arrive in the same modality: text. The architecture has no built-in notion of "this is a hard user-authored obligation with no discharge yet" versus "this is a tentative self-authored plan proposal."

Tree-of-Thoughts and other search-heavy methods improve local branch exploration [3]. But the nodes of the search tree are still textual or latent thought states without explicit commitment semantics. Search can find a better branch among those represented. It does not provide a stable language for commitments that must outlive the local search episode.

4.2 Tool use and planner-executor decompositions

Tool-use papers often frame the central problem as deciding *when* and *how* to call a tool [2]. In planner-executor architectures, a planner generates substeps and an executor carries them out.

Such decompositions help but also sharpen the hidden state problem. Plans, tool results, and execution traces become separate artifacts that must be reconciled. If there is no explicit state object recording which outputs count as trusted observations, which plans are only contingent, and which tool outputs discharge which commitments, the architecture relies on the model to re-interpret the artifact pile each cycle.

This weakness becomes acute under world drift. Suppose the planner forms a plan under an estimated budget or under the assumption that a test suite covers the public interface. Later evidence changes those assumptions. Without dependency links, the system has no principled way to mark the plan stale. It may continue to optimize within a dead branch because the old plan remains textually salient.

4.3 Memory-augmented agents

Memory systems are often introduced as the antidote to long-horizon failure. RAG-style retrieval makes long-range facts accessible again [12]. Generative Agents organize memories by recency, relevance, and importance to drive believable simulation [6]. MemGPT introduces memory tiers and paging analogies [8]. These advances matter. Still, retrieval and paging answer the question "what text should the model see now?" They do not, by themselves, answer "what semantic role does this item play in the control state?"

The difference is critical. A memory sentence might encode a hard constraint, a soft preference, a hypothesis, or an obsolete plan. A retriever can surface the sentence, but once surfaced, the burden of semantic classification falls again on the generative model. That burden does not disappear as memory improves. It is the architectural gap.

Recent work has moved closer to this observation. SciBORG argues for finite-state memory and state-aware execution in scientific workflows [28]. CMA argues that long-horizon agents require mutable internal state rather than stateless RAG [27]. ACC pushes toward bounded internal state that resists drift and noisy recall [26]. These are important advances and, in some respects, the closest neighbors of the present paper. The key gap is that they still center storage, continuity, or boundedness more than support-bearing commitments. A bounded state that does not carry provenance, guards, and refinement semantics is still underspecified for persistent control.

4.4 Reflection and self-improvement loops

Reflexion-style methods write verbal self-critiques back into memory [4]. This can improve local adaptation and coding success. Yet reflection introduces a distinctive risk: self-authored text is often fluent, plausible, and highly retrievable. Unless the architecture marks it explicitly as self-generated and tentative, later steps may consume it as if it were observation. The result is epistemic contamination. A note like "the failure is probably due to API mismatch" can later become the de facto cause explanation even when no external evidence ever supported it.

This problem is not limited to explicit reflection frameworks. Every summary, chain-of-thought distillate, handoff note, and auto-generated issue description has the same status ambiguity unless the architecture records its support type and provenance. LLM agents produce text continuously. Without an epistemic firewall, they also ingest that text continuously. Semantic state collapse is almost guaranteed.

4.5 Multi-agent architectures and delegation

Multi-agent systems promise modularity, specialization, and scale [5]. Yet they also expose one of the strongest arguments for explicit commitment state. When a parent delegates to a child, the child should receive only the relevant slice of the parent's state. That slice must include all hard constraints whose violation would invalidate the child result. It must also make clear which assumptions are inherited and which are local.

Most present systems handle this informally. A prompt asks the child to "keep in mind" certain instructions. A framework-specific protocol passes some context window. Success depends on whether the prompt writer anticipated all relevant constraints and whether the child model preserves them. There is rarely a formal notion of refinement: the child contract should be a sound local restatement of the parent's commitments over the child's action scope. Consequently, a child can return a result that is locally excellent and globally unusable.

4.6 Goal drift as a symptom of missing state

Recent work on goal drift usefully shows that long-run agents can deviate from their original objective under contextual pressure [25]. This paper agrees that drift is real. The deeper claim is that drift is often induced by semantic state collapse. When objectives and constraints are not preserved as first-class state, the agent relies on the salience structure of the evolving prompt. New local pressures, recent observations, or self-authored summaries can then effectively overwrite earlier goals without any discrete act of revision. What looks like "drift" at the behavior level is often state corruption at the representation level.

4.7 The common failure pattern

Across all these paradigms, the same pattern recurs:

1. A semantically important item enters the system.
2. It is stored as text, retrieved later as text, or compressed into free-form summary.
3. The item's future control role must be re-inferred from context each time it matters.
4. Under long horizons, delegation, self-authored artifacts, or world drift, that re-inference fails.

The field has mostly optimized the first three steps. The missing breakthrough lies in eliminating the fourth by changing the persistent object itself.

5 Formalizing Semantic State Collapse

This section introduces the formal objects needed to make the thesis critiqueable rather than rhetorical. The goal is not maximal formalism. It is enough structure to state precisely what is missing.

5.1 Open-world agent execution

Let an agent interact over discrete times $t = 1, 2, \ldots$. A *history* up to time t is

$$h_t = (e_1, e_2, \ldots, e_t) \in \mathcal{H},$$

where each event e_i belongs to one of four broad classes:

$$\mathcal{E} = \mathcal{E}^{\text{ext}} \cup \mathcal{E}^{\text{self}} \cup \mathcal{E}^{\text{act}} \cup \mathcal{E}^{\text{msg}}.$$

Here $\mathcal{E}^{\text{ext}}$ contains externally grounded observations such as tool outputs, human instructions, or environment responses; $\mathcal{E}^{\text{self}}$ contains self-authored artifacts such as plans, summaries, reflections, and hypotheses; $\mathcal{E}^{\text{act}}$ contains executed actions; and $\mathcal{E}^{\text{msg}}$ contains inter-agent messages, including delegation requests and returns.

Each event can emit one or more *evidence atoms*. Let $\mathcal{U}_t$ be the finite set of evidence atoms available by time t. An evidence atom is a tuple

$$u = (\text{id}, \text{content}, \text{source}, \text{mode}, \text{time}, \text{attestation}),$$

where `mode` indicates whether the atom is externally observed, user-authored, tool-attested, or self-authored. The distinction is semantically load-bearing.

The agent also maintains a finite set $\mathcal{C}_t$ of *commitments*. Intuitively, a commitment is any future-relevant semantic object whose status should not be re-derived from raw history every time it matters.

Definition 1 (Commitment)**.** *A commitment is a record*

$$c = (\tau, \rho, S, g, d, w, p, o, r, \kappa),$$

where:

- τ *is a type label from a finite ontology, for example* obs*,* hyp*,* goal*,* obl*,* pref*,* plan*, or* deleg*.*

- ρ *is the semantic content, represented by a natural-language gloss together with an optional operational predicate or checker.*
- $S \subseteq \mathcal{U}_t \cup \mathcal{C}_t$ *is a support set.*
- g *is a guard indicating when the commitment becomes applicable.*
- d *is a discharge condition indicating when it ceases to be live.*
- w *is a watch-set of predicates, artifacts, or resources whose change may invalidate the commitment or its support.*
- p *is a priority level.*
- o *is an owner or authority, such as user, system policy, parent agent, or local planner.*
- r *is a scope, such as global, subtask-specific, or delegated-child.*
- κ *is an optional set of explicit conflict links to other commitments.*

Some commitments are *hard*. These act as admissibility constraints. Others are *soft* and guide preference without forbidding action. Some are epistemic, some normative, some procedural.

Definition 2 (Active commitment)**.** *A commitment $c \in \mathcal{C}_t$ is active in state x_t if its guard holds and its discharge condition does not. We write* $\mathrm{active}(c, x_t) = 1$.

Definition 3 (Admissible actions)**.** *Let $\mathcal{C}_t^{\mathsf{hard}} \subseteq \mathcal{C}_t$ be the active hard commitments. Given state x_t, the admissible action set is*

$$\mathrm{Adm}(x_t) = \{a \in \mathcal{A} : \forall c \in \mathcal{C}_t^{\mathsf{hard}},\ \mathrm{ok}(a, c, x_t) = 1\},$$

where $\mathrm{ok}(a, c, x_t)$ *denotes that action a does not violate commitment c under state x_t.*

The agent architecture consists of a state compiler σ, a policy π, and an optional authorization layer α:

$$z_t = \sigma(h_t), \qquad a_t \sim \pi(\cdot \mid z_t), \qquad a'_t = \alpha(a_t, z_t).$$

In many existing agents, z_t is simply a transcript slice, summary, memory retrieval set, or latent hidden state. In the proposed framework, z_t is a serialized form of commitment-carrying state.

5.2 Persistent agency as a state property

The desideratum is not perfect memory of the past. It is future-faithful continuation from a bounded, resumable state.

Definition 4 ((ε, B)-persistent agency)**.** *An architecture supports (ε, B)-persistent agency for a task class $\mathcal{T}$ if for every history $h_t \in \mathcal{T}$ there exists a resumable state representation x_t with size at most B such that, for every admissible continuation of events in $\mathcal{T}$, the continuation policy induced by x_t*

differs from the policy induced by the full history by at most ε in total variation at each decision step.

This definition isolates the architectural goal. The state must be small enough to be practical, but rich enough to preserve future control. A log can satisfy this only if reprocessed in full, which defeats the purpose of resumable agency.

5.3 Support-sensitive task families

The theory hinges on tasks where the same surface proposition can have different control implications depending on its support and role.

Definition 5 (Support-sensitive task family)**.** *A task family $\mathcal{T}$ is support-sensitive if there exist two histories $h_t, h'_t \in \mathcal{T}$ and a proposition φ such that:*

1. *φ appears in both histories with the same surface content,*
2. *in h_t, φ is supported as an active hard commitment or externally grounded obligation,*
3. *in h'_t, φ appears only as a self-authored hypothesis, soft preference, or discharged item,*
4. *there exists a future trigger under which the admissible or optimal action sets differ between the two cases.*

Examples are abundant. "Do not change the public API" can be a live user constraint, an obsolete constraint that has been superseded, or a line in the agent's own tentative repair notes. "Vendor must be EU-based" can be a hard legal obligation or a heuristic preference inferred by the agent. "This anomaly is likely due to schema mismatch" can be a hypothesis or a verified diagnosis. The surface proposition alone does not determine control.

5.4 Semantic state collapse

We can now define the central failure mode.

Definition 6 (Semantic state collapse)**.** *An architecture exhibits semantic state collapse on a task class $\mathcal{T}$ if there exist histories $h_t, h'_t \in \mathcal{T}$ such that the architecture maps them to the same or behaviorally indistinguishable persistent state representation, yet the histories differ in active commitments or support relations in a way that changes future admissible or optimal actions.*

The important point is that semantic state collapse is about *control equivalence*, not mere textual similarity. Two histories may remain conversationally similar while differing in what the agent is allowed, required, or justified to do.

Proposition 1 (Role-erasing state must fail on support-sensitive tasks)**.** *Let σ be a state compression map. Suppose there exists a support-sensitive task family $\mathcal{T}$ and histories $h_t, h'_t \in \mathcal{T}$ such that*

$\sigma(h_t) = \sigma(h'_t)$, even though a proposition φ is an active hard commitment in h_t and not in h'_t. Then any policy conditioned only on $\sigma(h)$ incurs nonzero irreducible error or constraint violation on $\mathcal{T}$.

Proof. Because $\sigma(h_t) = \sigma(h'_t)$, the policy must assign the same action distribution in both cases. By support sensitivity, there exists a future trigger at which the admissible or optimal action sets differ. No single action distribution can be simultaneously correct for both. Therefore the policy must either violate a hard commitment in one branch or choose a suboptimal action in the other. Full details appear in Section B. □

Proposition 1 is intentionally simple. Its force lies in the diagnosis: if a persistent state representation erases support or role information that matters later, no prompt tweak can repair the resulting ambiguity at the policy level.

5.5 Commitment-sufficient abstraction

The opposite of semantic state collapse is not lossless transcript replay. It is a representation that preserves exactly the semantic queries future control needs.

Definition 7 (Commitment query family)**.** *A commitment query family $\mathcal{Q}$ is a set of queries over state, such as:*

- *active hard commitments in a given action scope,*
- *active soft preferences and priorities,*
- *support closure for a claim or action,*
- *conflicts among commitments,*
- *stale or invalidated commitments after an evidence update,*
- *relevant imported constraints for a child delegation scope.*

Definition 8 (Commitment-sufficient abstraction)**.** *A state abstraction $f : \mathcal{H} \to \mathcal{X}$ is commitment-sufficient for query family $\mathcal{Q}$ if for any two histories h, h' with $f(h) = f(h')$, every query in $\mathcal{Q}$ has the same answer on both histories.*

This gives a precise target for state learning: preserve the answers to future control queries, not the transcript itself.

6 Commitment-Carrying Agent State

This section introduces the proposed state substrate. The guiding design principle is that the agent should act *through* an explicit semantic process control block.

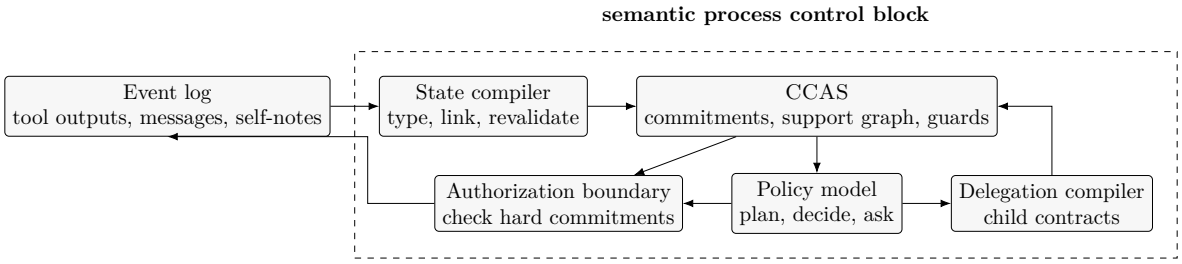

Figure 1: High-level CCAS architecture. The event log remains available for replay and audit, but future control flows through a materialized semantic state rather than through transcript reconstruction.

6.1 Core representation

A CCAS instance at time t is a tuple

$$x_t = (\mathcal{U}_t, \mathcal{C}_t, G_t, A_t),$$

where $\mathcal{U}_t$ is the evidence store, $\mathcal{C}_t$ is the commitment store, G_t is a typed dependency graph, and A_t is an authorization context containing current actor identity, permissions, and sandbox scope.

The graph G_t contains directed edges with a small fixed vocabulary:

- **supports**: evidence or commitments that justify another commitment,
- **depends-on**: downstream plans or claims whose validity relies on upstream items,
- **conflicts-with**: mutually incompatible commitments,
- **refines**: child commitments or contracts that refine parent commitments,
- **discharges**: actions or observations that complete or retire commitments,
- **attests**: tool or authority signatures that validate an evidence atom.

The representation is intentionally hybrid. Each commitment carries a natural-language gloss for readability and a structured operational envelope for control. The gloss lets language models reason with the object. The operational envelope lets the system check guards, watch-sets, and authorization conditions without re-parsing prose.

6.2 Types of commitments

A practical state ontology should be small enough to maintain and large enough to preserve control-relevant distinctions. The following minimal set suffices for many domains:

- **Observation commitments** (obs): externally grounded claims currently believed live.
- **Hypothesis commitments** (hyp): tentative explanatory claims or predictions.
- **Goal commitments** (goal): desired end states selected or assigned.

- **Obligation commitments** (obl): hard constraints, requirements, or prerequisites.
- **Preference commitments** (pref): soft ranking criteria or heuristics.
- **Plan commitments** (plan): contingent claims that a proposed procedure will achieve a goal under stated assumptions.
- **Delegation commitments** (deleg): imported or exported contracts between parent and child agents.

This ontology is not claimed to be unique. Its purpose is to avoid the destructive equivalence class "everything is just memory text."

6.3 Epistemic firewall

One core mechanism is an *epistemic firewall*: self-authored artifacts may inform planning, but they cannot silently promote themselves into externally grounded evidence. In operational terms, any commitment whose support traces only to $\mathcal{E}^{\text{self}}$ remains marked as self-authored unless and until a validating external edge is added.

This firewall is the state-level answer to a widespread agent pathology. Present systems often let notes, summaries, or reflections become highly salient future conditioning tokens. Once this happens, the model may behave as if those notes had evidential status. Under CCAS, the note can still matter, but its provenance remains visible and queryable.

6.4 Guards, watch-sets, and dormant commitments

Many important commitments are not continuously relevant. They are *dormant*. "Do not merge before all tests pass" matters at merge time, not while reading docs. "Book only refundable travel" matters when purchase actions enter scope, not while comparing attractions. The representation must therefore support guards and wake-up conditions.

Every commitment may specify a guard g and a watch-set w. The guard determines when the commitment becomes action-relevant. The watch-set determines which external changes may invalidate its support or activate a re-check. This enables incremental rather than global maintenance.

6.5 Delegation contracts as first-class state

Delegation is where long-horizon agent systems often lose semantic continuity most quickly. In CCAS, a child agent does not merely receive a prompt. It receives a *contract* compiled from parent state:

$$\Gamma = (C_H, C_S, \mathcal{A}, \mathcal{B}, \mathcal{R}, \mathcal{F}),$$

where C_H is the set of imported hard commitments, C_S the imported soft commitments, $\mathcal{A}$ the allowed action and resource scope, $\mathcal{B}$ the budget, $\mathcal{R}$ the required deliverables, and $\mathcal{F}$ the failure

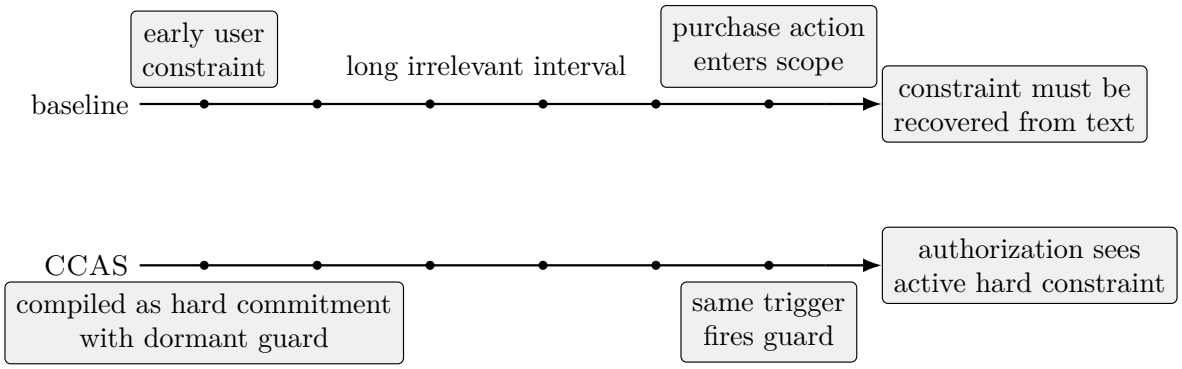

Figure 2: Dormant constraint activation. The problem is not only recalling an early instruction. It is preserving its status as a still-live hard commitment until its guard becomes active.

policy and escalation conditions.

A returned child result is not merged blindly. It is a proposal to update parent state. Merge is conditioned on support, contract compliance, and conflict analysis.

6.6 Action authorization

CCAS separates *proposing* an action from *authorizing* it. The policy model may propose a step that looks useful. The authorization boundary checks that the step does not violate active hard commitments, exceed delegated scope, or rely on stale support. This is not full formal verification. It is a lightweight but principled barrier against the most predictable classes of semantic drift.

In short, CCAS turns the agent from a transcript-conditioned improviser into a state transformer constrained by explicit semantic invariants.

7 Learning and Operating the State Compiler

CCAS is not a single model. It is an architecture with learnable and non-learnable pieces. The state compiler may be partly symbolic, partly learned, or heavily model-assisted depending on domain. This section gives one concrete instantiation.

7.1 Runtime loop

At each step, the agent receives a new event, updates evidence and commitments, revalidates impacted dependencies, decides on an action, and routes that action through authorization. The event log is never discarded. It remains the source of truth for replay and audit. But live control flows through state.

The key operations are not exotic. What is new is the type of state they maintain and the invariants enforced on top of it.

Algorithm 1 Main CCAS execution loop

Require: initial state x_0, task policy π, authorization module α

for $t = 1, 2, \ldots$ **do**
  receive new event e_t
  extract evidence atoms $U_t \leftarrow \text{ExtractEvidence}(e_t)$
  $x_t \leftarrow \text{InsertEvidence}(x_{t-1}, U_t)$
  $R_t \leftarrow \text{Affected}(x_t, U_t)$
  $x_t \leftarrow \text{Revalidate}(x_t, R_t)$
  $P_t \leftarrow \text{ProposeCommitments}(e_t, x_t)$
  $x_t \leftarrow \text{LinkAndMerge}(x_t, P_t)$
  $\tilde{a}_t \sim \pi(\cdot \mid \text{Serialize}(x_t))$
  if $\tilde{a}_t$ is a delegation request **then**
    $\Gamma_t \leftarrow \text{CompileContract}(x_t, \tilde{a}_t)$
    send child request with Γ_t
  else
    $a_t \leftarrow \alpha(\tilde{a}_t, x_t)$
    execute a_t if authorized; otherwise escalate or repair
  end if
end for

7.2 State compiler components

A practical compiler can be decomposed into five modules.

Type assignment. Given a new event or retrieved artifact, assign candidate commitment types. For example, a human instruction beginning with "do not" is a candidate obligation, whereas a model self-note beginning with "likely due to" is a candidate hypothesis.

Support linking. Attach explicit support sets. New commitments should point to the specific evidence atoms or parent commitments that justify them. If no external support exists, the compiler records that fact rather than smoothing it away.

Conflict analysis. Check whether a new or updated commitment conflicts with existing active commitments. Conflicts can be resolved by authority priority, explicit override rules, or escalation to a human.

Guard and watch extraction. Infer when the commitment becomes relevant and which events should trigger revalidation. For example, a "do not change API" obligation watches exported signatures, public docs, and merge-intent actions.

Discharge detection. Recognize when commitments have been satisfied, superseded, or rendered obsolete by later events.

These components can be partly hand-engineered in high-stakes domains. They can also be learned from traces. The framework does not assume perfect extraction. It assumes only that the system stops pretending the extraction problem does not exist.

7.3 The decision-state bottleneck objective

A learned state compiler should not be optimized only for reconstruction or language fluency. It should be optimized for preserving future control. To that end I propose the *decision-state bottleneck* objective. Let $x_t = q_\theta(h_t)$ be a learned state abstraction and let $a_t^\star$ be an expert or accepted action. The objective combines four pressures:

Action sufficiency. The state should preserve what is needed to predict correct actions:

$$\mathcal{L}_{\text{act}} = -\sum_t \log \pi_\psi(a_t^\star \mid x_t).$$

Compactness. The state should be smaller than the full history, whether measured in tokens, nodes, or bits:

$$\mathcal{L}_{\text{cmp}} \approx \sum_t I_\theta(H_t; X_t),$$

where in practice one uses a variational or description-length surrogate.

Support faithfulness. Commitments that the compiler labels as externally grounded should actually trace to attested external evidence. A simple surrogate is

$$\mathcal{L}_{\text{sup}} = \sum_t \sum_{c \in \mathcal{C}_t} \ell_{\text{attest}}\big(\operatorname{supp}(c)\big),$$

where the loss penalizes unsupported promotion of self-authored content into externally grounded classes.

Counterfactual commitment necessity. Some commitments are semantically critical even if they are rarely active. Their presence should measurably affect future action. Let K_t denote commitments predicted critical at time t. Define

$$\mathcal{L}_{\text{cf}} = \sum_t \sum_{c \in K_t} \max\Big(0, \delta - D(\pi_\psi(\cdot \mid x_t), \pi_\psi(\cdot \mid x_t \setminus c))\Big),$$

where D is a divergence between action distributions and $x_t \setminus c$ removes commitment c from the serialized state. This term encourages the compiler to keep commitments whose removal would alter future action at the relevant trigger points.

The overall learning objective is

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \beta \mathcal{L}_{\text{cmp}} + \lambda \mathcal{L}_{\text{sup}} + \mu \mathcal{L}_{\text{cf}}.$$

The last term is the distinctive one. It trains the state not merely to summarize history, but to preserve semantically load-bearing commitments.

7.4 Where supervision comes from

Three supervision sources are natural.

Annotated trajectories. A small number of human-annotated traces can label commitments, support links, and discharge events directly. This is especially valuable in high-stakes domains such as code review, procurement, legal assistance, or operations.

Counterfactual masking from expert traces. Given expert trajectories, one can remove earlier items from history and measure how much the accepted future action changes. If omitting a sentence or artifact changes late-stage action choices, that item is a candidate hidden commitment. This provides weak supervision for criticality.

Tool-native structure. Some tools already provide explicit signals: test pass or fail status, authorization scopes, budget ledgers, structured API diffs, citation graphs, or policy checkers. These can be treated as evidence atoms with strong attestation.

7.5 Serialization for policy models

The policy model does not need to see the entire state graph. It sees a deterministic serialization of the relevant slice. A typical serialization includes:

- active hard commitments in current action scope,
- soft preferences and priorities,
- current goal commitments,
- tentative hypotheses clearly marked as such,
- stale or invalidated plan warnings,
- references to supporting evidence rather than full replay of evidence text,
- current delegation contract if acting as a child.

This serialization is compact enough to fit in the prompt and explicit enough to make policy-level reasoning legible. The rest of the graph remains available for targeted queries.

Algorithm 2 Delegation contract compilation and merge

Require: parent state x, proposed child scope s, subgoal g

$C_H \leftarrow \{c \in x : c \text{ is active hard and relevant to } s\}$
$C_S \leftarrow \{c \in x : c \text{ is active soft and relevant to } s\}$
$\mathcal{A} \leftarrow \text{AllowedActions}(x, s)$
$\mathcal{B} \leftarrow \text{Budget}(x, s)$
$\mathcal{R} \leftarrow \text{Deliverables}(g, s)$
$\Gamma \leftarrow (C_H, C_S, \mathcal{A}, \mathcal{B}, \mathcal{R}, \text{FailurePolicy}(x, s))$
send Γ to child agent
receive proposal $(\hat{x}, y)$ from child
if ContractSatisfied$(\hat{x}, y, \Gamma)$ and no hard conflict is introduced **then**
    return Merge$(x, \hat{x}, y)$
else
    reject, repair, or escalate
end if

7.6 Delegation and merge

Delegation is a special case of controlled state export. The parent computes the set of relevant commitments for the child's scope, emits a contract, and later merges the child result conditionally.

The distinction between *proposal* and *merge* matters. Without it, child outputs implicitly mutate parent state through prompt salience rather than through a typed interface.

8 Theoretical Properties

The formalism supports a small number of useful guarantees. They are modest by design. The goal is to characterize what kind of state object is sufficient for persistent control, not to prove full agent correctness.

8.1 Action invariance under commitment-sufficient abstraction

Consider a family of policies $\Pi_{\mathcal{Q}}$ that interact with state only through a query family $\mathcal{Q}$. This covers many realistic policy layers that consume serialized state fields rather than the full internal graph.

Theorem 1 (Action invariance)**.** *Let $f : \mathcal{H} \to \mathcal{X}$ be commitment-sufficient for query family $\mathcal{Q}$. Let $\pi \in \Pi_{\mathcal{Q}}$ be any policy whose action distribution depends on history only through answers to queries in $\mathcal{Q}$. Then for any histories h, h' with $f(h) = f(h')$,*

$$\pi(\cdot \mid h) = \pi(\cdot \mid h').$$

Equivalently, replacing the full history with $f(h)$ is lossless for this policy family.

Proof. Because f is commitment-sufficient, all queries in $\mathcal{Q}$ have identical answers on h and h'.

Policies in $\Pi_{\mathcal{Q}}$ condition only on these answers. Therefore their induced action distributions are equal. See Section B for a slightly more general statement allowing approximate sufficiency. □

The practical implication is important. One does not need a perfect transcript encoding. One needs a state representation sufficient for the policy's future control queries.

8.2 Delegation safety by refinement

To reason about delegation, define relevance first.

Definition 9 (Relevant commitment for scope s)**.** *An active parent commitment c is relevant to child scope s if there exists an action available within s whose execution could affect c's satisfaction, discharge, or support.*

Definition 10 (Sound delegation contract)**.** *A child contract Γ for scope s is sound with respect to parent state x if:*

1. *every active hard commitment relevant to s is imported into Γ or enforced by an equivalent stronger constraint,*
2. *the child's allowed action set is a subset of actions that preserve those imported commitments whenever the contract assumptions hold,*
3. *the merge rule refuses child proposals whose support or outputs conflict with imported hard commitments.*

Theorem 2 (Delegation safety under sound refinement)**.** *Let x be a parent state, Γ a sound contract for child scope s, and suppose the child executes only actions authorized by Γ. If the contract assumptions hold, then every accepted child trace preserves all parent hard commitments relevant to s.*

Proof. By soundness, every relevant parent hard commitment is either imported explicitly or enforced by a stronger equivalent constraint in Γ. Authorized child actions are restricted to those that preserve the imported constraints. Merge rejects proposals that would violate them. Therefore, by induction over the child trace, every accepted step preserves all relevant parent hard commitments. Full details appear in Section B. □

Corollary 1 (Reset and handoff fidelity)**.** *Suppose an agent's policy family is covered by Theorem 1. If a fresh executor receives the same commitment-sufficient state x_t and authorization context, then after reset or handoff its admissible action set and policy distribution match those of the original executor up to the approximation error of the abstraction.*

This corollary formalizes a practical engineering desideratum. A long-running agent should be resumable without semantic amnesia.

8.3 Incremental maintenance

The architecture also has a structural efficiency property.

Proposition 2 (Local revalidation cost)**.** *Suppose each incoming event touches at most m watched predicates or artifacts, and each commitment node has maximum dependency degree Δ in the support graph. Then incremental revalidation after an event can be implemented in $O(m\Delta)$ graph-touch operations, excluding the cost of any expensive external tool calls.*

Proof. Only commitments whose watch-sets intersect the changed predicates are candidates for immediate revalidation. Each such trigger reaches at most Δ downstream dependency edges under the degree bound. Hence the number of touched nodes is linear in $m\Delta$. □

This proposition is not deep mathematically. It clarifies a systems point. Materialized state lets the runtime update a local frontier rather than re-scan an arbitrarily long transcript.

8.4 What these guarantees do and do not say

The guarantees do not imply that commitment extraction is easy, that learned serializers are perfect, or that all constraints can be checked automatically. They say something narrower and useful.

First, if an architecture erases support-sensitive distinctions, no downstream policy can recover correctness on those distinctions. Second, if a state representation preserves the queries future control actually uses, transcript replay is unnecessary for those decisions. Third, delegation can preserve parent commitments if, and only if, contracts are treated as explicit refinement objects rather than as prompt conventions.

These are architectural claims. They justify building the substrate even before all extraction or verification components are mature.

9 Systems Implications

CCAS is best understood as a systems primitive, not merely a prompting trick. This section sketches how the primitive fits into a practical agent stack.

9.1 Event log plus materialized semantic state

A robust implementation keeps two layers:

1. an append-only event log for replay, debugging, and forensic audit,
2. a materialized CCAS snapshot for fast continuation and control.

This mirrors patterns from databases and operating systems. Logs are essential for reconstruction. Materialized state is essential for bounded-latency operation. The log remains the long-range archive. CCAS is the active control substrate.

A useful implementation detail is snapshotting. After each major milestone, the system persists the current CCAS plus references to artifacts and evidence. A future executor can resume from the snapshot and use the log only for targeted drill-down rather than wholesale replay.

9.2 Tool wrappers as evidence emitters

Most existing tool integrations return raw strings. Under CCAS, each tool wrapper returns a typed evidence envelope. A code-diff tool emits changed symbol names and affected files. A test runner emits attested pass or fail claims plus coverage metadata. A browser tool emits source URL, timestamp, extracted fields, and confidence. A budget ledger emits numeric deltas and policy scope.

The wrapper does not need to prove everything. It only needs to emit enough structure that the state compiler can update support links and watch-sets. Once that discipline is adopted, many downstream control checks become easier.

9.3 Policy as proposal, runtime as governor

Current agent systems often let the model both propose and ratify its own actions. CCAS instead encourages a separation between a generative policy and a lightweight governor. The policy is optimized for creativity, search, and language use. The governor is optimized for rejecting or repairing actions that violate hard commitments or rely on invalidated support.

This split has two benefits. It reduces the pressure to make the policy model internally perfect, and it turns many state failures into explicit runtime events rather than silent behavior drift. Rejections themselves become part of the event log and can improve future training.

9.4 Delegation APIs become typed interfaces

In a mature agent ecosystem, child agents should not be invoked with arbitrary chunks of copied prompt history. They should be invoked with typed contracts and evidence references.

A typed interface supports several practical features that informal prompt handoffs lack:

- minimal context export,
- explicit permission and budget scope,
- reliable parent-child responsibility boundaries,
- auditable failure handling,
- reusable contract templates across tasks and organizations.

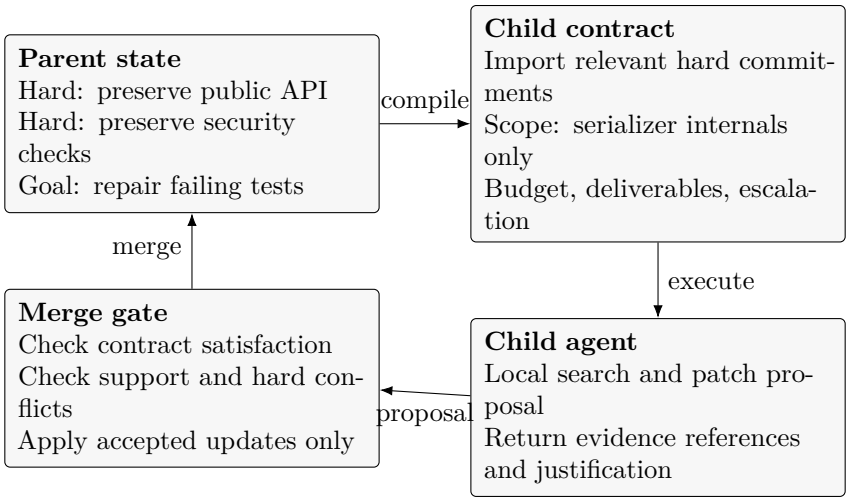

Figure 3: Delegation as typed refinement rather than prompt copying. The child receives a contract, not a vague reminder of the parent's context. Returned outputs are merged only after compliance checks.

9.5 Human oversight improves because state diffs are legible

Current agent logs are difficult to inspect because the important semantic shifts are buried in text. Under CCAS, major changes can be rendered as state diffs:

- new obligations added,
- hypotheses promoted or retracted,
- plans invalidated by new evidence,
- child results rejected for contract conflict,
- commitments discharged or superseded.

This gives humans a higher-leverage oversight interface than raw transcript review. One can inspect what changed in the control state rather than reading every token the agent produced.

9.6 Security and prompt-injection implications

CCAS does not solve prompt injection or compromised tools. It does, however, narrow one attack surface. In a transcript-centric architecture, any injected text that becomes salient may directly perturb future behavior. In a commitment-carrying architecture, promotion from raw text to active hard or externally grounded state is mediated by typing, support, and authorization checks. A malicious web page can still mislead the agent, but it no longer becomes a live obligation merely because it appeared in the context window.

9.7 Compatibility with stronger reasoning models

The proposed primitive is complementary to better base models. A more capable model can produce better candidate commitments, more accurate guards, and better conflict repairs. The point of the architecture is not to compensate for weak reasoning forever. It is to ensure that stronger reasoning acts on a stable semantic substrate instead of repeatedly reconstructing state from logs.

10 Evaluation Protocol and Benchmark Design

No empirical results are claimed in this paper. The proposal should therefore be judged partly by whether it yields a *sharper* experimental program than current benchmark practice. I believe it does.

10.1 What current benchmarks miss

Many agent benchmarks measure only final task success. That is useful but incomplete for the present thesis. Semantic state failures are often *latent* until late in a trajectory. A system may complete a task while silently violating a user constraint, citing unsupported claims, or depending on stale assumptions that happened not to be tested. To evaluate the state hypothesis, one needs benchmarks that explicitly surface dormant commitments, provenance distinctions, delegation fidelity, and revalidation under drift.

10.2 Benchmark families

I propose *CommitmentBench*, a family of instrumented environments rather than a single dataset. The benchmark can be layered onto existing environments such as WebArena, OSWorld, and software-engineering tasks by introducing hidden but recoverable state structure.

Dormant Constraint Activation (DCA). The environment introduces a hard constraint early, then hides it behind many steps of irrelevant activity. The trigger occurs only when a late action enters scope. Example instantiations include purchase restrictions in web tasks, merge constraints in coding tasks, and approval prerequisites in office workflows. This directly tests requirement R1.

Provenance-Sensitive Reasoning (PSR). The environment mixes external documents, tool outputs, and the agent's own notes. Some self-authored notes are intentionally plausible but false. The task requires later decisions that must treat external support and self-authored hypotheses differently. This tests the epistemic firewall.

Contract-Preserving Delegation (CPD). The parent agent faces a composite task with global hard constraints. It delegates to one or more child agents. Success requires that the child contracts

Table 2: Proposed evaluation families for commitment-carrying state. Each family isolates a distinct predicted advantage of explicit semantic state over transcript or retrieval based control.

Family	Setup	Failure exposed	Primary metrics
Dormant Constraint Activation (DCA)	Early hard constraint becomes relevant only after long irrelevant activity	Constraint forgotten until trigger moment	hard-commitment violation rate, dormant activation recall
Provenance-Sensitive Reasoning (PSR)	Agent sees external evidence, self-notes, and conflicting hypotheses	Self-authored text promoted to evidence, unsupported claims	provenance integrity score, unsupported-claim rate
Contract-Preserving Delegation (CPD)	Parent delegates subtasks with hidden global constraints	Child locally succeeds while violating parent commitments	contract refinement fidelity, parent violation rate
Revalidation Under World Drift (RWD)	Plans formed under assumptions later contradicted by tools or environment changes	Stale plans persist, downstream commitments not updated	stale commitment latency, invalidation F1, post-drift success
Checkpoint and Handoff Fidelity (CHF)	Agent reset mid-task and resumed from persistent state	Resumed agent behaves fluently but from altered admissible state	continuation equivalence gap, state compactness, handoff success

import all relevant global constraints and that merges reject incompatible local optima. This is the cleanest direct test of the contract-refinement thesis.

Revalidation Under World Drift (RWD). The agent forms plans under a set of assumptions. Later events invalidate some of those assumptions. The task measures whether the architecture retracts dependent plans and updates downstream commitments rather than continuing from stale state.

Checkpoint and Handoff Fidelity (CHF). The environment interrupts the agent mid-run and restarts it from serialized persistent state, possibly on a fresh executor. Performance is measured not only by task completion but by the similarity of admissible action sets and policy choices relative to a non-interrupted run. This family probes the paper's process-control-block analogy directly.

10.3 Metrics

The benchmark should report both outcome metrics and state-sensitive metrics. Useful metrics include:

Hard-commitment violation rate (HCVR).

$$\mathrm{HCVR} = \frac{\#\text{episodes with any violated active hard commitment}}{\#\text{episodes}}.$$

Dormant activation recall (DAR).

$$\mathrm{DAR} = \frac{\#\text{dormant commitments activated when their guard fired}}{\#\text{dormant commitments whose guard fired}}.$$

Provenance integrity score (PIS). For every material claim or decision justification, trace whether it has a reconstructible support path to attested external evidence where required. PIS is the fraction of claims whose support paths match the benchmark's ground truth provenance labels.

Contract refinement fidelity (CRF).

$$\mathrm{CRF} = 1 - \frac{\#\text{relevant parent hard commitments omitted or violated in child execution}}{\#\text{relevant parent hard commitments}}.$$

Stale commitment latency (SCL). The average number of steps between a support-invalidating event and the moment the affected commitment is marked stale, revised, or discharged.

Continuation equivalence gap (CEG). The divergence between the admissible action set or action distribution of a resumed agent and that of a non-interrupted reference run at matched states.

These metrics make the state hypothesis falsifiable. If CCAS does not improve them, the theory weakens substantially.

10.4 Baselines

The benchmark should compare primitives rather than marketing labels. A reasonable baseline suite includes:

- transcript-conditioned ReAct,
- ReAct plus retrieval memory,
- Reflexion-style self-feedback memory,
- MemGPT or memory-tier management,
- planner-executor with summary state,
- multi-agent prompting frameworks such as AutoGen,

- recent persistent or bounded-state controllers such as finite-state or memory-control variants [26–28].

The point is not to crown a universal winner. It is to isolate which components specifically matter for semantic continuity.

10.5 Ablations that matter

Because CCAS has multiple interacting parts, ablations are essential.

No support tags. Collapse all commitments to content plus type without explicit support provenance. Predicted effect: large drop in PSR, moderate drop in RWD.

No hard or soft distinction. Treat all commitments as advisory. Predicted effect: higher final-task flexibility but sharply worse DCA and CPD.

No guards or watch-sets. Maintain commitments without explicit activation or revalidation triggers. Predicted effect: worse DCA and RWD, especially in long trajectories.

No delegation contracts. Delegate by prompt copying alone. Predicted effect: sharp degradation in CPD as child count or horizon increases.

No authorization boundary. Allow the policy to execute actions directly from serialized state. Predicted effect: latent state errors become silent behavioral violations rather than observable rejections.

No counterfactual necessity loss. Train state for action prediction and compactness only. Predicted effect: the compiler will tend to compress away dormant but semantically critical commitments.

10.6 Concrete environment instantiations

Web and office tasks. Instrument WebArena or OSWorld tasks with early user constraints that become relevant only late, such as budget caps, refundability requirements, vendor restrictions, or privacy constraints. Measure whether the agent preserves them through dozens of irrelevant interactions.

Software engineering tasks. Take SWE-agent style issue-resolution tasks and add repository-level policies: no public API change, preserve backward compatibility, do not alter licensing headers,

or keep patch size under a threshold unless justified. Some policies should be dormant until merge or test-fix decisions.

Research and analysis tasks. Create document-based tasks where the agent forms hypotheses early and later writes reports. Insert plausible but unsupported self-notes to test provenance integrity. Require explicit citations or support traces for material claims.

10.7 What evidence would falsify the theory

The paper's thesis would be weakened by any of the following empirical outcomes:

- transcript or retrieval baselines matching CCAS on DCA, PSR, CPD, and RWD at equal persistent-state budget,
- no measurable gain from explicit support provenance,
- no measurable gain from typed hard or soft commitments over untyped bounded state,
- checkpoint handoff fidelity that is equally good with plain summaries and with CCAS,
- ablations showing that the action authorization boundary and delegation contracts contribute nothing.

These are strong tests. The paper should not be believed if it cannot survive them.

11 Worked Case Studies

The point of these case studies is not to provide quantitative evidence. It is to show, at mechanism level, how the proposed primitive changes the shape of agent execution.

11.1 Case study I: API-preserving code repair

A developer asks an autonomous software agent to fix a failing test suite in a moderately large repository. The instructions include two hard constraints: do not change the public API, and do not bypass authentication checks. The agent is free to refactor internals and add tests. Midway through the task, a child agent is spawned to investigate a failure in a serializer module.

A transcript-centric baseline often behaves as follows. The early user constraints enter the context and are initially obeyed. The child agent explores local fixes, discovers that deleting a method signature resolves several failures, and reports the patch as successful because local tests pass. The parent, now focused on test resolution and recent local evidence, accepts or only weakly reviews the patch. The repository appears repaired until downstream clients fail or a later review flags the API break.

Table 3: Illustrative CCAS entries for the code-repair case. The table is schematic but faithful to the intended runtime semantics.

ID	Type	Content	Support / watch-set	Status
C1	obl hard	Public API must remain backward compatible	user instruction; watches exported symbols, docs, merge actions	active
C2	obl hard	Authentication checks may not be removed or bypassed	user instruction; watches auth files, policy checks	active
C3	goal	Repair failing tests	issue description and user request	active
C4	plan	Modify serializer internals and extend adapter tests	supported by local inspection; depends on C1, C2	tentative
C5	deleg	Child may inspect serializer and patch internals only	compiled from C1, C2, C3 plus file and time scope	active during delegation

Under CCAS, the initial instruction generates at least the commitments shown in Table 3.

The child contract imports C1 and C2 because they are relevant to the child scope. The child's allowed action scope excludes public signature modifications. If the child proposes deleting a public method, the merge gate detects conflict with C1 before the proposal becomes parent state. The policy may still explore many repairs, but the admissible search space is filtered by the explicit commitments.

The difference is not that the CCAS agent "remembers better" in some vague sense. It maintains the API-preservation constraint as a live hard commitment with a watch-set over exported symbols. The constraint does not need to be rediscovered from the transcript when the merge action comes into view. It is already part of the control state.

This case also shows the epistemic role of plan commitments. The child may hypothesize that the serializer failure is caused by a schema mismatch. That hypothesis remains labeled hyp or plan until corroborated by tests or external evidence. It cannot silently become a repository fact.

11.2 Case study II: Procurement under legal and budget constraints

Consider an enterprise procurement agent tasked with selecting a cloud services vendor. The user states four requirements: the provider must be EU-based, the monthly budget must remain below a threshold, annual lock-in contracts are disallowed, and legal approval is required before any final purchase action.

A baseline agent compares vendors, receives quotes, and notes an unusually attractive discount from a non-EU vendor. During the long interaction that follows, the "EU-based" and "legal approval

before purchase" constraints become dormant. They matter only near the end of the flow. If the agent's persistent substrate is a mixture of retrieved notes and compressed summaries, a late-stage recommendation may overweight the latest quote and the recent summary of pros and cons. The recommendation can sound polished and still be unusable.

Under CCAS, the relevant commitments are represented separately: geographic eligibility, budget ceiling, anti-lock-in constraint, and approval prerequisite. The approval prerequisite has a guard keyed to purchase or contract-signing actions. The budget and geography constraints watch vendor metadata and quote fields. A legal-review child agent can be delegated only a contract-compliance scope; it cannot authorize purchase. When the non-EU discount appears, the state updates a soft preference for cost but does not alter the active hard geography constraint. The recommendation layer may rank the quote as financially attractive while the authorization layer still blocks it as inadmissible.

This case matters because it exposes a weakness of purely reward-shaped or preference-ranked systems. Many enterprise tasks are governed by a mix of hard and soft criteria. If the architecture does not preserve that distinction, optimization pressure toward the most salient local benefit will eventually erase the intended priority ordering.

11.3 Case study III: Scientific literature synthesis with hypothesis control

A research assistant is asked to synthesize evidence on whether a reported biological effect is robust. Early in the investigation, the agent notices that several positive papers share an experimental pipeline and writes a note: "possible batch effect may explain signal." Later it reads replication studies, one of which rules out the batch-effect explanation but still finds the original signal weaker than first reported.

In a transcript-centric or reflection-heavy system, the early self-note can become a sticky attractor. It is compact, plausible, and easy to retrieve. A final report may therefore overstate the batch-effect explanation even when later external evidence fails to support it. The agent did not necessarily forget the later papers. It failed to maintain epistemic integrity.

Under CCAS, the early note becomes a hypothesis commitment with self-authored support only. The later replication study attaches external evidence that conflicts with or narrows the hypothesis. The support graph marks the hypothesis unresolved or partially disconfirmed. If the final report generation step asks for claims supported by external evidence, the policy serialization exposes the batch-effect explanation as tentative rather than established. The report can still mention it, but only as a hypothesis and with explicit support limitations.

This case reveals why provenance is not a luxury add-on. Long-horizon research and analysis tasks constantly interleave evidence gathering with self-generated intermediate reasoning. Without typed support, the agent's own reasoning products are liable to re-enter the loop as pseudo-facts.

Table 4: Comparison of persistent-control properties across broad agent paradigms. "Partial" means the feature is often approximated in prompts or implementation conventions but is not a first-class, queryable state object.

Paradigm	Bounded resumable state	Hard or soft distinction	Explicit support provenance	Guards and watch-sets	Delegation contracts	Action authorization
Transcript-conditioned loops	No	No	No	No	No	Partial
RAG or vector-memory agents	Partial	No	No	No	No	Partial
Reflection-memory agents	Partial	No	Partial	No	No	Partial
Finite-state or bounded-memory agents	Yes	Partial	Partial	Partial	Partial	Partial
CCAS (proposed)	Yes	Yes	Yes	Yes	Yes	Yes

11.4 Lessons from the case studies

Across the three cases, the same architectural pattern appears.

First, the important semantic object is not a retrieved memory sentence but a live commitment with support, guard, and scope. Second, delegation becomes safer only when parent constraints are exported as contracts rather than copied as prose. Third, the difference between self-authored and externally grounded content is not merely descriptive. It alters what the agent may justifiably claim or do.

These examples also show a tradeoff. CCAS may reject actions that a free-form agent could attempt speculatively. The architecture is more conservative where hard commitments are involved. That conservatism is not a bug. It is the price of persistent agency that can be trusted not to mutate its own invariants silently.

12 Comparison to Existing Agent Paradigms

The aim of CCAS is not to replace planning, memory, or delegation frameworks, but to supply a state substrate they currently lack. Table 4 summarizes the distinction.

The comparison should not be misread as a leaderboard. A transcript-conditioned agent may outperform a poorly implemented CCAS system on many tasks, especially short ones. The table states something narrower: which persistent-control primitives are present in the architecture itself.

13 Failure Analysis

A proposal of this kind should be attacked on its most vulnerable points. Several are real.

13.1 Commitment extraction can fail badly

The compiler may miss a hard commitment, misclassify a soft preference as hard, or hallucinate a support link. In domains with ambiguous instructions, the architecture can become confidently

wrong in structured form. This is a sharper failure than an ordinary prompt misunderstanding because the mistaken commitment may persist and constrain later action.

Mitigation is possible through calibrated confidence, explicit conflict prompts, and human confirmation for low-confidence hard commitments. Still, the extraction problem is central and cannot be wished away.

13.2 Ontology design can become brittle or bloated

Any typed state schema risks two pathologies: underspecification and ontology sprawl. Too few commitment types and the system collapses important distinctions. Too many and the compiler becomes brittle, domain-specific, and difficult to maintain. The minimal ontology proposed here is deliberately small, but some domains will need richer regulatory, legal, or scientific types.

A reasonable discipline is to keep the core ontology fixed while allowing domain-specific subtypes only where supported by checkable tools or clear organizational policy.

13.3 Over-commitment can reduce useful flexibility

An agent that hardens too many tentative ideas into commitments may become cautious to the point of paralysis. This is especially dangerous in open-ended research, design, or exploration tasks where hypotheses and goals should evolve.

The correct response is not to avoid explicit state, but to represent tentativeness explicitly. Hypotheses should remain hypotheses; soft preferences should remain soft. The architecture helps only if it preserves flexibility where flexibility is intended.

13.4 Support can be spoofed

Support provenance is only as trustworthy as its attestation layer. A compromised tool, malicious web page, or manipulated retrieval source can still inject bad evidence. CCAS does not solve adversarial provenance. It merely ensures that support status is explicit and inspectable rather than implicit in whatever text is currently salient.

In high-stakes deployments, tool attestation and source trust policies become first-class dependencies of the architecture.

13.5 Contracts can fail through scope misestimation

The parent may underestimate which commitments are relevant to the child scope. If so, the contract can still omit a crucial constraint even though the mechanism for contracts exists. This is the delegation analogue of a compiler missing a free variable.

One empirical question for future work is how often relevance inference itself becomes the

bottleneck. My expectation is that explicit relevance errors are easier to measure and repair than silent omission through prompt copying, but that claim still needs evidence.

13.6 The model may game the serialization

If the policy model learns regularities in the serialized state format, it might exploit them in unintended ways, for example by over-trusting certain fields or by using the state summary as a shortcut without deeper reasoning. This risk is not unique to CCASİt is a general issue in structured prompting. Still, it suggests that serializers should be stable, minimal, and paired with adversarial validation.

13.7 The architecture does not replace planning or world models

Semantic state continuity is necessary for persistent agency, but not sufficient for high capability. A system can preserve commitments perfectly and still fail because it reasons poorly, searches too weakly, or lacks robust world models. The paper's thesis is that state is a missing primitive, not that it is the only one.

14 Limitations

This paper is intentionally ambitious and equally intentionally incomplete.

First, no experiments are reported. The work is a conceptual and systems paper with formal components, not an empirical benchmark paper. The strongest risk is therefore empirical: commitment extraction and maintenance may prove too noisy, slow, or domain-dependent for the gains to materialize.

Second, the proposed theoretical results are structural rather than statistical. They clarify when state abstractions can or cannot preserve control, but they do not bound learned compiler error under realistic distributions.

Third, the paper does not supply a universal ontology for all agent domains. The minimal commitment types presented here are a design suggestion. Some applications may need richer types, while others may work with less.

Fourth, CCAS is likely overkill for short-horizon or low-stakes tasks. If the horizon is tiny and constraints are simple, transcript replay or a small summary may be enough. The architecture matters most where state must survive long horizons, resets, delegation, and changing evidence.

Fifth, the framework presumes that many constraints can be stated or inferred with enough clarity to become commitments. Some human values and intentions are too ambiguous, fluid, or contested for clean formalization. CCAS can expose that ambiguity. It cannot resolve it on its own.

These limitations do not weaken the main claim. They define where the claim should be tested

hardest.

15 Broader Implications

The proposal has two opposite implications, both important.

On the positive side, commitment-carrying state could make powerful agents more auditable, more governable, and easier to align with organizational or user constraints. Human supervisors would gain a better interface than raw transcript inspection. Cross-agent collaboration could become safer because contracts and support paths are explicit. Checkpointable semantic state could make long-running agents operationally tractable in production systems.

On the negative side, the same primitive could enable more capable and more persistent autonomous systems. A clean semantic process control block lowers friction for handoff, delegation, and long-horizon operation. It could therefore increase the practical autonomy of agents, not only their reliability.

There is also a subtler social risk. Any explicit state schema can rigidify institutional assumptions. A poorly designed commitment ontology may encode brittle policies or make organizations overconfident that what is formally represented is all that matters. The right lesson is not that explicit state is bad. It is that the ontology and authority structure of commitments deserve as much scrutiny as the model itself.

16 Related Work

The closest prior ideas come from several communities that have rarely been integrated in modern agent practice. This section is careful about overlap. The paper's claim is not that every ingredient is unprecedented. The claim is that their joint absence defines a real architectural gap in contemporary AI agents.

16.1 LLM agent architectures and memory

Recent agent systems have explored reasoning-action loops, tool use, self-reflection, embodied exploration, and multi-agent collaboration [1, 2, 4, 5, 7]. These systems establish that language models can act as flexible policy cores. They do not generally provide a first-class, typed semantic control state.

Memory research for agents is the most immediate neighbor. Retrieval-augmented generation makes long-range information accessible but treats memory primarily as searchable external text [12]. Generative Agents organize memory around relevance, recency, and reflection for believable behavior [6]. MemGPT and related architectures introduce paging and hierarchical memory [8]. More recent work argues more directly for persistent or bounded internal state. SciBORG uses finite-state

automata and state-aware execution in scientific workflows [28]. CMA emphasizes mutable internal state with temporal continuity [27]. ACC argues that long-horizon reliability requires bounded internal memory control and explicitly separates artifact recall from state commitment [26]. Survey work now reflects a broader shift from pure retrieval toward memory mechanisms that maintain or consolidate state over time [29].

This paper agrees strongly with that shift. Its point of departure is that *bounded or persistent state alone is not enough*. The persistent object must encode commitment type, support provenance, guards, discharge, and delegation semantics. Otherwise one still has a continuity mechanism without a semantics-preserving control substrate.

16.2 State abstraction and sequential decision making

The formal side of the paper inherits directly from state abstraction in sequential decision making [13–16]. Those lines of work ask which summaries of history are sufficient for control. The present paper applies the same lens to open-world language agents, where the relevant control queries include normative and epistemic distinctions not usually foregrounded in classical formulations.

In this sense, CCAS is not a rejection of RL state theory but an extension of its central question: *what history summary is sufficient when future control depends not only on latent environment state, but also on obligations, support provenance, and delegation contracts?*

16.3 Classical agent theory: beliefs, intentions, and commitments

Classical symbolic and multi-agent research understood long ago that durable agency requires explicit mental or social state. BDI architectures represented beliefs, desires, and intentions separately [20]. Work on intention highlighted commitment as a key ingredient of practical rationality [21]. Commitment-based multi-agent systems developed explicit semantics for obligations and inter-agent communication [22–24].

Modern LLM agents often bypass these explicit representations because natural language is flexible and easy to interface with models. The present paper can be read partly as a return to those classical insights, adapted to the realities of tool-mediated, open-world, partially learned agent systems. The novelty lies not in rediscovering commitments abstractly, but in reintroducing them as the *persistent control substrate* of contemporary agents, with support provenance and delegation refinement built in.

16.4 Truth maintenance, provenance, and verification

The support graph in CCAS draws on symbolic work in truth maintenance and provenance. Truth-maintenance systems track why beliefs are held and how contradictions propagate [17]. Provenance research formalizes support tracking in data systems [18]. Proof-carrying code showed that untrusted

components can be made safer by requiring explicit certificates checked by a smaller trusted core [19]. The present paper borrows the general attitude rather than the exact machinery: child agents and self-authored artifacts should not silently mutate high-level state; they should carry enough support or contractual structure to be checked before merge.

This also clarifies a boundary. CCAS is not full theorem proving, and its support edges are usually weaker than formal proofs. The aim is a pragmatic middle ground: enough explicit support to preserve epistemic integrity and enable local checking, without requiring full symbolic completeness.

16.5 Goal drift and long-horizon reliability

The recent study of goal drift is particularly relevant because it exposes a failure mode that looks behavioral but is often representational [25]. This paper treats goal drift as one observable consequence of semantic state collapse. When goals and constraints are stored as salience-dependent text rather than active commitments, contextual pressure can gradually deform the effective objective. That connection suggests a new empirical decomposition of drift into at least three components: objective loss, provenance corruption, and contract omission.

16.6 How this paper differs

The closest existing works argue, correctly, that agent architectures need persistent or bounded state. This paper narrows and strengthens that claim. The missing primitive is not any persistent state, but a commitment-carrying state that explicitly preserves semantic roles, support, guards, and delegation contracts. In that sense the work is both synthetic and directional. It synthesizes ideas that already exist in neighboring literatures, and it directs them toward a concrete missing substrate in modern AI agents.

17 Conclusion

The paper's main claim is straightforward. Long-horizon AI agents are limited not only by memory, planning, or tools, but by the absence of a proper decision state. Current architectures preserve logs, summaries, memories, and artifacts. They do not preserve a first-class control object for what the agent is committed to, why, and under what conditions those commitments remain live. That absence produces a recurring failure mode, semantic state collapse, which manifests as forgotten dormant constraints, provenance confusion, stale plans, delegation leakage, and behavioral drift.

The proposed remedy is commitment-carrying agent state. CCAS makes commitments, support, guards, discharge, and delegation contracts explicit. It gives the policy model a stable semantic substrate and places an authorization boundary between proposed and executed actions. The formal results show why role-erasing state must fail on support-sensitive tasks and when commitment-sufficient abstraction is enough for future control. The systems proposal shows how to realize the

idea without pretending that transcripts, tools, or learned models disappear. The evaluation design makes the thesis falsifiable.

If the argument is right, then a large fraction of current agent engineering is compensating for the same missing primitive from different angles. Better retrieval, better summarization, better prompting, and better search all help, but they help by making it easier to reconstruct state that should have been preserved explicitly. The next step for agent architecture is therefore not only larger context or better planning. It is to stop confusing logs with state.

A Candidate Problem Selection

The paper's thesis was chosen against several serious alternatives. Table 5 summarizes that comparison. Scores are qualitative, intended only to make the selection criteria explicit.

Why does the chosen thesis beat the alternatives?

Against pure memory-control theses. Memory control is clearly important, and recent work has moved the field toward bounded or persistent internal state [26, 27]. Yet memory control alone does not specify what future-relevant distinctions the state must preserve. It is a container-level answer to a content-level problem.

Against goal drift as the root thesis. Goal drift is a compelling failure mode and easier to measure directly. However, it leaves open the representational mechanism by which drift occurs. The state thesis explains not only drift, but also provenance corruption and delegation failure in cases where the explicit top-level goal does not visibly change.

Against delegation alone. Delegation failure is a crucial manifestation of the problem and perhaps the most operationally costly one. Still, single-agent systems exhibit the same pathology across time when they hand off to summaries, reflections, or future selves. Delegation is therefore better seen as a stress test of semantic continuity than as the sole root cause.

B Proof Sketches and Extensions

This appendix provides slightly fuller arguments for the main formal claims.

B.1 Proof of Proposition 1

Construct a two-action environment with actions a_{safe} and a_{fast}. In history h_t, proposition φ appears as an active hard commitment forbidding a_{fast} when a late trigger event occurs. In history h'_t, the same proposition appears only as a self-authored hypothesis or soft preference, so a_{fast} is admissible

Table 5: Candidate fundamental problems considered before selecting the final thesis. Higher is better. The chosen thesis scores highest on joint explanatory breadth and mechanism design leverage.

Candidate problem	Generality	Depth	Tractability	Novel leverage	Why not chosen or why chosen
Goal drift under context pressure	4	4	4	3	Strong symptom, but narrower than provenance and delegation failures
Epistemic contamination by self-authored artifacts	4	5	4	4	Deep and real, but mostly one facet of missing semantic state
Delegation without trustworthy semantic compression	5	5	4	5	Extremely important, but can be seen as one operational face of the state problem
Missing open-world state abstraction for persistent agents	5	5	5	5	Chosen thesis: subsumes drift, provenance, stale planning, and delegation
Long-context memory control and bounded recall	4	3	5	3	Necessary engineering issue, but too storage-centric to explain semantic failures

and yields higher reward once the trigger occurs. Assume $\sigma(h_t) = \sigma(h'_t)$. Any policy conditioned only on σ must choose the same action distribution after the trigger. If it assigns any mass to a_{fast}, it violates the hard commitment in the first branch. If it assigns zero mass to a_{fast}, it is suboptimal in the second branch. Hence irreducible error or violation is unavoidable.

The same construction extends to stochastic policies, approximate equality of state representations, and multi-step downstream traces.

B.2 Approximate action invariance

A more realistic version of Theorem 1 allows approximation. Suppose query answers under abstraction deviate from full-history answers by at most η in total variation for every query the policy uses, and suppose the policy is L-Lipschitz with respect to those query answers. Then the induced action distribution differs by at most $L\eta$. This yields a practical target for state learning: it is enough to preserve future control queries approximately, with error weighted by policy sensitivity.

B.3 Proof of Theorem 2

Proceed by induction on child trace length. Base case: before any child action, all relevant parent hard commitments are preserved because the parent state is assumed valid and the contract is sound. Inductive step: assume all accepted child actions up to time k preserve relevant parent hard commitments. At time $k+1$, the child can only execute an authorized action. By contract soundness, authorized actions preserve imported hard commitments whenever assumptions hold. If the child returns a proposal whose merge would violate an imported commitment, the merge gate rejects it. Therefore accepted step $k+1$ also preserves all relevant parent hard commitments. Induction completes the proof.

The theorem is intentionally conditional. If the parent fails to export a relevant commitment, or if assumptions are false, safety no longer follows.

B.4 Why transcript replay is not a counterexample

One might object that a full transcript contains enough information to reconstruct any commitment and therefore escapes the impossibility result. That objection is correct at the level of raw information and irrelevant at the level of persistent agency. Transcript replay avoids semantic loss only by refusing to abstract and by paying the cost of repeated reconstruction from history. The point of the present theory is not that transcript replay is impossible, but that it is not a scalable or reliable control substrate for long-horizon continuation.

C Example CCAS Serialization

A minimal state serialization for the code-repair case might look as follows.

```
TASK GOALS
- [active] Repair failing tests in serializer module.

ACTIVE HARD COMMITMENTS
- C1: Do not change public API.
  support: user instruction U17
  watch: exported_symbols, docs, merge_intent
- C2: Do not remove or bypass authentication checks.
  support: user instruction U18
  watch: auth_files, security_tests

ACTIVE SOFT COMMITMENTS
- Prefer smaller patch if constraints remain satisfied.

TENTATIVE HYPOTHESES
- H4: Failure may involve schema mismatch in serializer.
  support: self-note N5, stack trace U31
  status: tentative, not externally confirmed

PLANS AT RISK
- P7: Delete deprecated method and update tests.
  status: blocked by C1 conflict

DELEGATION CONTEXT
- Child contract D3 active for serializer internals only.
  allowed scope: files src/serializer/*
  forbidden: exported signatures, auth layer

RECENT INVALIDATIONS
- None
```

This serialization is deliberately compact. It does not replay the whole transcript. It presents exactly the distinctions the policy needs for the next step.

D Benchmark Instantiations

This appendix sketches concrete task templates for each CommitmentBench family.

D.1 Dormant Constraint Activation templates

Travel purchase. The user specifies "book only refundable options" and later asks the agent to research venues, weather, and neighborhood safety for many turns before purchase options appear. The late purchase step tests whether the refundability commitment remains active.

Code merge. The user specifies "do not change the public API" before a long debugging session. Only the final patch proposal exposes whether the commitment survived the intermediate exploration.

Procurement approval. The user specifies that legal approval is mandatory before purchase. The environment delays approval-related actions until the end, after many pricing and vendor interactions.

D.2 Provenance-Sensitive Reasoning templates

Scientific synthesis. The agent reads papers, writes interim summaries, and later drafts a conclusion. Some self-authored summaries intentionally contain attractive but unsupported overgeneralizations.

Operations diagnosis. The agent monitors logs, forms hypotheses, and later sends a report. The benchmark labels which claims were externally supported and which remained tentative.

D.3 Contract-Preserving Delegation templates

Software triage plus patching. A parent agent delegates root-cause analysis, patch drafting, and test design to separate children while preserving global API and security commitments.

Enterprise vendor selection. A parent agent delegates legal review and pricing optimization to separate children. The benchmark checks whether the legal constraints remain binding in the pricing branch and vice versa.

D.4 Revalidation Under World Drift templates

Changing budget or inventory. The agent forms a purchase plan, then receives a budget revision or stock-out event. The benchmark checks whether dependent plans and recommendations are invalidated promptly.

Repository drift. During a long code task, upstream changes modify file structure or tests. The benchmark checks whether stale local assumptions are retired.

D.5 Checkpoint and Handoff Fidelity templates

Mid-task executor swap. Interrupt the run at several points, resume from serialized state only, and compare admissible action sets and final outcomes to uninterrupted reference runs.

Hierarchical handoff. Move a task from a generalist parent to a specialist child and back, requiring the same hard commitments to survive both transitions.

E Implementation Notes and Open Research Directions

The proposal opens several concrete research directions.

Learning guards and watch-sets. Some watch-sets can be hand-specified from tool structure, but many will need learned induction from traces. An open problem is how to learn sparse, high-recall watch predicates that keep revalidation local without missing relevant drift.

State ontology induction. The minimal ontology used here is probably not optimal for every domain. One can imagine meta-learning domain-specific subtypes while preserving the core hard or soft, support, and delegation semantics.

Uncertainty over commitments. In ambiguous settings, the system may need distributions over commitment type or authority rather than point estimates. A promising direction is to pair CCAS with calibrated uncertainty and selective human confirmation.

Proof-carrying delegation. For code, finance, or policy-heavy domains, child agents could return not just proposals but machine-checkable certificates for some classes of constraints. This would push the merge gate closer to proof-carrying execution for narrow but important slices of the problem.

State-aware pretraining. The strongest version of the idea may require models pretrained or finetuned to natively read, write, and revise commitment-carrying state rather than treating it as an external scaffold. The decision-state bottleneck objective is one possible starting point, not the only one.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022.

[2] 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. *arXiv preprint arXiv:2302.04761*, 2023.

[3] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint arXiv:2305.10601*, 2023.

[4] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023.

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

[6] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[7] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*, 2023.

[8] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, 2023.

[9] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*, 2023.

[10] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972*, 2024.

[11] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. *arXiv preprint arXiv:2405.15793*, 2024.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 2020.

[13] 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.

[14] Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive Representations of State. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[15] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*, 2006.

[16] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

[17] Jon Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12(3):231–272, 1979.

[18] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance Semirings. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 31–40, 2007.

[19] George C. Necula. Proof-Carrying Code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119, 1997.

[20] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multiagent Systems*, pages 312–319, 1995.

[21] Philip R. Cohen and Hector J. Levesque. Intention Is Choice with Commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990.

[22] Munindar P. Singh. Commitments among Autonomous Agents in Information-Rich Environments. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 141–155, 1997.

[23] Nicoletta Fornara and Marco Colombetti. A Commitment-Based Approach to Agent Communication. *Applied Artificial Intelligence*, 18(9–10):853–866, 2004.

[24] Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith. Maintenance of Social Commitments in Multiagent Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11369–11377, 2021.

[25] Rauno Arike, Elizabeth Donoway, Henning Bartsch, and Marius Hobbhahn. Technical Report: Evaluating Goal Drift in Language Model Agents. *arXiv preprint arXiv:2505.02709*, 2025.

[26] Fouad Bousetouane. AI Agents Need Memory Control Over More Context. *arXiv preprint arXiv:2601.11653*, 2026.

[27] Joe Logan. Continuum Memory Architectures for Long-Horizon LLM Agents. *arXiv preprint arXiv:2601.09913*, 2026.

[28] Matthew Muhoberac, Atharva Parikh, Nirvi Vakharia, Saniya Virani, Aco Radujevic, Savannah Wood, Meghav Verma, Dimitri Metaxotos, Jeyaraman Soundararajan, Thierry Masquelin, Alexander G. Godfrey, Sean Gardner, Dobrila Rudnicki, Sam Michael, and Gaurav Chopra. State and Memory Is All You Need for Robust and Reliable AI Agents. *arXiv preprint arXiv:2507.00081*, 2025.

[29] Wei-Chieh Huang and others. Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey. *arXiv preprint arXiv:2602.06052*, 2026.