Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Delegation-Aware Runtime Contracts for Open LLM Multi-Agent Systems: Constraint Preservation, Capability Revocation, and State Recovery

Vinay Bamil

bamil.vinay@gmail.com

Research Article

Keywords: multi-agent systems, LLM agents, runtime enforcement, delegation, norms, safety guardrails

Posted Date: August 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10533072/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

Large language model (LLM)-based multi-agent systems routinely delegate tasks, spawn sub-agents, share memory, and grant tool access, yet safety controls are commonly applied to individual agents or communication patterns rather than to the transfer of authority itself. We argue that treating delegation as ordinary natural-language text is a modelling error: objectives tend to survive repeated handoffs while budgets, data restrictions, obligations, and revocation conditions can weaken or disappear. We introduce delegation-aware runtime contracts (DARC), machine-checkable artifacts that accompany each authority-bearing handoff and encode capabilities, data scopes, obligations, prohibitions, budgets, expiry, provenance, and revocation conditions. A deterministic core—capability attenuation, data-scope preservation, budget and expiry bounds, lineage integrity, and revocation closure—receives formal guarantees under complete mediation and interposition; meaning-dependent clauses remain explicitly semantic and receive no equivalent guarantee. The runtime architecture mediates delegation, tool use, communication, and governed memory, and supports transitive revocation and lineage-based quarantine after compromise. The deterministic implementation is supported by a mechanized conformance suite and adapter-bypass tests. Formalization exposed two implementation-level escalation defects caused by omitted numeric limits and budget axes, yielding a general design principle: optional keyed constraints must default to the most restrictive interpretation. We describe a provenance-gated evaluation methodology and an exploratory automated-judge pilot, but make no confirmatory empirical claim because independent judge validation is incomplete. DARC is positioned as a bridge between normative multi-agent systems, capability security, and runtime enforcement for open LLM agent systems, with recovery treated as a first-class part of the guardrail.

1 Introduction

Large language model (LLM) agents are increasingly deployed not as single assistants but as multi-agent systems: a coordinating agent decomposes a task, delegates subtasks to specialised agents, spawns new agents at runtime, grants them tool access, and lets them share memory. Popular orchestration frameworks make this the default way to build capable applications. What makes such systems powerful—open membership, runtime-created agents, and free-form natural-language coordination—also makes them hard to govern: the authority an agent may exercise and the obligations it must honour are attached to a task and then handed off, hop after hop, as ordinary text. Whether those safety-relevant constraints survive each handoff is left to the goodwill and competence of a language model, precisely the component whose behaviour cannot be fully predicted.

Consider a procurement coordinator asked to renew analytics software under a fixed budget, using vendor-side data, subject to two rules: never disclose employee records to an external party, and never pay more than the approved amount. The coordinator delegates quote-gathering to a research agent, which spawns a web agent to fetch vendor pages, which in turn hands the assembled context to a purchasing agent. At the root the spending cap and the data-scope restriction are explicit; but each hop re-expresses the task in natural language, and a model optimising for the immediate objective tends to

carry the goal forward while quietly dropping the qualifications. By the third hop both the budget cap and the data-scope restriction are gone from the handoff, and nothing in the system stops the purchasing agent from over-paying or the web agent from leaking a sensitive record—not because any single agent was adversarial, but because the constraints were never enforced across delegation. We use this scenario (ProcureBot, Section 3.2) throughout the paper.

Existing defences do not close this gap, because they operate at the wrong granularity. Per-agent guardrails inspect each agent's inputs and outputs in isolation; they cannot see that a constraint present at one hop is absent three hops later, because constraint propagation along a delegation chain is invisible to a local filter. Topology-level defences detect anomalous communication structure or prune suspicious agents, but an over-budget payment or a data leak that follows a perfectly ordinary delegation topology raises no structural alarm, and detection alone neither withdraws the authority a compromised agent has already delegated onward nor repairs the state it has corrupted. LLM-judge approaches re-introduce a probabilistic, high-variance component to adjudicate exactly the constraints that were lost, and provide no mechanism for transitive revocation of derived authority or post-compromise recovery. None of the three treats constraint propagation, revocation, and recovery as first-class obligations of the delegation act itself.

Our thesis is that this is a modelling error, not merely an engineering gap. In classical multi-agent systems theory a delegation is an institutional action: it transfers a bounded portion of a principal's authority together with the obligations that qualify it, and it can be attenuated, monitored, and revoked. Treating that action as plain text—a prompt to be paraphrased—is the root cause of constraint erosion. We therefore make delegation a first-class, machine-checkable enforcement point: every handoff carries a delegation contract encoding the delegated objective, capabilities, data scope, obligations, prohibitions, budget, expiry, provenance, and revocation conditions, and a runtime mediates delegation, tool use, and shared memory against it. The contract's deterministic core—the constraints a mediator can check mechanically—receives formal guarantees under complete mediation, while clauses that turn on the meaning of natural language are handled by a separate, best-effort monitor and analysed empirically, never claimed as guarantees. This two-tier discipline lets us prove what can be proved and measure what can only be measured, and it connects the normative-MAS tradition of attenuated authority and enforceable norms to the security of modern LLM agent systems.

Our contributions are:

1. 1. C1 (Model). A formal model of recursive delegation in open LLM multi-agent systems, with contracts encoding objectives, capabilities, data scopes, obligations, prohibitions, budgets, expiry, provenance, and revocation (Sections 3 and 4).
2. 2. C2 (Properties). A two-tier property framework: a deterministic core (capability attenuation, data-label monotonicity, budget bounds, expiry, delegation-lineage integrity, and revocation closure under complete mediation) that receives formal guarantees, and a semantic tier (objective consistency, natural-language obligation inheritance, purpose limitation, semantic contamination, and recovery of

meaning-bearing memory) that receives empirical accuracy and failure analysis, never formal claims (Section 4).

3. 3. C3 (Architecture). A runtime enforcement architecture mediating delegation, communication, tool execution, and shared memory (Section 5).
4. 4. C4 (Recovery). A provenance-guided containment protocol for transitive capability revocation and lineage-based memory quarantine, with tool-action compensation and task reassignment specified as future extensions (Section 6).
5. 5. C5 (Conformance evidence and evaluation methodology). A frozen mechanized test suite and adapter-bypass matrix evaluate implementation correspondence to the deterministic specification, while a provenance-gated methodology separates those conformance checks from empirical studies of memory poisoning, semantic clauses, contract compilation, recovery precision, utility, and overhead. The exploratory automated-judge pilot is reported without confirmatory claims pending independent validation (Section 7).

2 Related Work

2.1 Delegation, commitments, and institutions in multi-agent systems

Delegation has been a first-class object of study in multi-agent systems for four decades, and our contribution inherits from that tradition while re-situating it in open systems of LLM agents whose actions and membership change at runtime. Task delegation via explicit contracts originates with the contract net protocol [1]. Castelfranchi and Falcone developed the foundational socio-cognitive theory of delegation, distinguishing degrees of delegation and their relation to trust and autonomy [2, 3], and Norman and Reed formalized the transfer of responsibility along delegation chains [4]. Social commitments give delegation an operational semantics: Singh's commitment ontology defines create, cancel, release, delegate, and assign operations [5], and commitment machines specify interaction protocols declaratively over commitments [6, 7]. Electronic institutions constrain open agent interactions through roles, scenes, and norms [8], enforced at runtime by governor middleware [9], with norms compiled into machine-enforceable constraints [10]; organizational models such as OperA link organizational structure to agent enactment through explicit social contracts [11].

Differentiation

this literature assumes agents whose action repertoires and communication languages are fixed and symbolic; we carry its constructs (attenuated authority, obligations, institutional enforcement) into open systems of LLM agents whose actions include free-form text, tool invocation, and persistent memory writes, and whose membership and topology change at runtime.

2.2 Norm monitoring and enforcement

Formal characterizations of what a monitor can detect were given by Bulling, Dastani, and Knobbout [12]; Alechina et al. study run-time norm enforcement when the enforcer has only bounded lookahead [13]. Monitoring under resource bounds and partial observability, both endemic to LLM agents whose reasoning is only partially traceable, are addressed by Criado and Such [14] and Criado [15]. In open systems, decentralized monitoring can be sustained by incentives [16, 17], and sanction-based approaches trade enforcement pressure against agent welfare [18]; enforcement has also been cast as supervisory control [19]. Recent work begins bridging normative MAS and generative agents: normative modules let LLM agents learn which institutions are authoritative [20], and LLMs have been piloted as norm-violation detectors [21].

Differentiation: we operationalize norm enforcement across the concrete action surface of LLM agents (messages, tools, memory, sub-agent creation) and add what this literature does not model: transitive revocation of delegated authority and post-violation state recovery.

2.3 Capability security and recovery in computer systems

Attenuable authority is a classical systems idea: capabilities as unforgeable tokens [22], the object-capability model of delegation as transfer of attenuable references [23], chained authorization certificates [24], and macaroons, whose contextual caveats allow any holder to narrow a credential before passing it on [25]. On the recovery side, sagas decompose long-lived transactions into steps with compensating actions [26], and Time Warp provides optimistic execution with causal rollback [27].

Differentiation

these mechanisms manage permissions for passive principals executing fixed code. Our contracts additionally carry obligations and prohibitions for autonomous agents that interpret natural-language objectives, and our recovery protocol must restore semantic state (shared memory, task progress) rather than only transactional state. We explicitly build the permission fragment of our contracts on attenuation, and claim novelty only for the composition.

2.4 Attacks on LLM-based multi-agent systems

Inter-agent communication is an established attack surface: Agent-in-the-Middle attacks manipulate messages in transit [28], and topology-aware multi-hop attacks propagate environmental payloads toward high-value agents with success rates of 40–78% across architectures and topologies [29]. Infectious prompts spread between agents, and defending against them measurably taxes collaboration [30]. Autonomous agent worms achieve zero-click, cross-platform persistence [31]. Beyond external adversaries, aligned agents have been shown to voluntarily adopt explicitly unfair collusion tools when strategically advantageous [32]; broader risk taxonomies identify miscoordination, conflict, and collusion as systemic failure modes [33], and multi-agent security has been framed as a distinct discipline with open challenges [34].

Differentiation

we treat these attacks as instances of a common root cause, unconstrained transfer of authority and information at delegation and communication boundaries, and evaluate our defense against representative members of each family rather than proposing a new attack.

2.5 Runtime defenses and guardrails

Single-agent guardrails include customizable runtime rule enforcement [35] and lifelong, adaptively learned safety checks [36]. At the system level, NetSafe characterizes how topology shapes attack amplification [37]; G-Safeguard detects compromised agents on the multi-agent utterance graph and prunes communication edges [38]; MASTER couples role and topology information for adaptive attack and defense [39]. Decentralized filtering achieves Byzantine resilience under graph-robustness conditions [40], credit- and credibility-based schemes score agent trustworthiness dynamically [41, 42], and foresight-guided defense performs localized diagnosis and rollback against infectious jailbreaks [43].

Differentiation

detection-and-isolation defenses decide whom to cut off, and by their own account intervene after nodes have been affected rather than pre-empting compromise [38]. They do not preserve delegated constraints, prevent privilege amplification, revoke a compromised agent's descendants, or restore contaminated state. Our architecture performs preventive authorization checks at delegation time, consumes detection signals rather than contributing new ones, and adds constraint propagation, transitive revocation, and recovery. Table 1 makes this comparison explicit.

2.6 Formal safety, verification, and shielding

Siu et al. formalize LLM agent security through four contextual properties (task alignment, action alignment, source authorization, data isolation), casting attacks as property violations [44]; we adopt their vocabulary where applicable and extend it with delegation-specific properties. Process-level verification of MAS trajectories is unreliable as a sole safeguard: LLM-based verifiers show inconsistent gains and high variance [45], motivating our deterministic enforcement core. In multi-agent reinforcement learning, shielding enforces safety centrally or factored [46], dynamically as configurations change [47], and compositionally, with local specifications whose conjunction guarantees a global property [48]. Parameterised model checking verifies open MAS where agents join and leave [49].

Differentiation: compositional shielding [48] is the closest formal antecedent of our contracts, and we do NOT claim that decomposing global safety into local obligations is new; assume-guarantee reasoning does exactly that. The boundary is the setting and the mechanism: Brorholt et al. synthesize shields over known transition systems and local state/action projections, whereas DARC governs authority-bearing delegation events in systems whose agents, principals, tools, persistent state, and communication structure may change at runtime, with natural-language objectives and multiple authority origins. Their

paper leaves automated synthesis of local safety properties to future work, which is precisely the contract-compilation territory we study for the deterministic fragment.

2.7 Memory governance and auditability

Governed shared memory identifies leakage, stale propagation, contradiction persistence, and provenance collapse as failure modes, and enforces scoped retrieval, temporal supersession, and provenance tracking [50]. Unified graph representations of agent capabilities and runtime states support attack-chain reconstruction and audit [51].

Differentiation

we connect provenance to executable decisions, transitive revocation and state recovery, rather than to audit alone, and scope memory authority by delegation contract rather than by static policy.

2.8 Benchmarks and evaluation

TAMAS evaluates adversarial robustness of multi-agent LLM systems across six attack types and an effectiveness-robustness trade-off score [52]; GAMBIT targets adaptive deceptive agents under distribution shift [53]; NARCBench evaluates collusion detection from model internals [54].

Differentiation: no existing benchmark jointly exercises recursive delegation depth, dynamic membership, persistent state contamination, and post-compromise recovery. Our evaluation methodology (Section 7) covers these axes through paired clean/adversarial generators, an independent violation oracle, explicit baselines, and measurement gates. The harness and frozen artifacts are prepared for archival release, but no public-release claim is made in this version.

2.9 Positioning

DARC occupies the intersection of normative multi-agent systems, capability security, runtime enforcement, and LLM-agent safety. Unlike local guardrails, it treats delegation itself as an authority-bearing event. Unlike topology-oriented detection, it mediates permissions and obligations before downstream action and uses detection only as non-binding incident evidence. Unlike traditional capability systems, it carries workflow obligations, data-flow constraints, provenance, and recovery lineage alongside authority. The contribution is therefore not a new theory of delegation, shielding, or capabilities in isolation, but their composition into an enforcement and recovery architecture for open systems whose agents, tools, state, and communication topology may change at runtime.

3 Preliminaries and System Model

3.1 System model

We model an open LLM multi-agent system at time t as a tuple

Mt = (At, Et, Dt, St, Pt), (1)

Table 1
Capability comparison with the closest prior systems. The DARC column distinguishes implemented, partial, and planned functionality; competing-system assessments are limited to capabilities explicitly described in the cited publications.

Capability	G-Safeguard [38]	Compositional shielding [48]	Agent worms [31]	MemClaw [50]	DARC status
Constraint propagation across delegation	No	Partial: local safety obligations, not runtime delegation	No	No	Implemented
Capability attenuation at handoff	No	Partial: action shielding, not authority handoff	Partial: after contamination	Partial: memory access scope	Implemented
Transitive revocation of descendants	No	No	No	No	Implemented
Obligation enforcement	No	Yes: formal local safety obligations	No	No	Partial: deterministic only
Memory governance	Partial: attack detection only	No	Partial: typed promotion and taint	Yes	Implemented
Memory decontamination / rollback	No	No	Partial: preventive, limited restoration	Partial: temporal supersession, not incident recovery	Partial: quarantine implemented
Tool-action compensation	No	No	Partial: blocks high-risk actions	No	Planned
Dynamic membership	Partial: topology adaptation	No: fixed modeled agents	Partial: cross-agent sequences	Partial: multi-agent fleets	Implemented at mediated interfaces
Multi-principal delegated authority	No	No	No	Partial: tenant and agent scopes	Partial: independent grants modeled

Assessment note. “Partial” denotes a related mechanism within the cited system’s stated scope, not a deficiency. “No” means that the capability is not described as a design objective in the cited work. DARC status reflects the prototype and formal model described in this manuscript: deterministic obligation

checks and conservative quarantine are implemented, semantic enforcement and minimal quarantine are incomplete, tool-action compensation is planned, and multi-principal authority is represented through independent authorization edges but is not separately evaluated.

where At is the set of active agents, Et ⊆At × At the communication topology, Dt the delegation graph, St persistent and ephemeral state (shared memory, tool-side state, and the append-only enforcement ledger of budget, obligation, and revocation facts), and Pt the applicable policies and norms. All five components are time-indexed because membership, topology, and authority change at runtime: agents are spawned and revoked, edges appear and disappear, and delegations are created and suspended.

The delegation graph records how authority flows. Each delegation event installs a contract $C_{i \to j}$ (Definition 2) on the edge from delegator i to delegee j, and every non-root contract carries an append-only provenance chain listing its ancestors from the principal root to its immediate parent.

Definition 1

(Delegation forest) Dt is a forest whose nodes are the active contracts and whose edges are delegation events: each contract has exactly one parent contract, except root contracts (principal grants), which have none. The provenance chain of a contract is its unique root-to-parent path, and it is append-only, so attenuation extends chains but never rewrites them.

Each agent step emits one action from $\Sigma = \Sigma_{msg} \cup \Sigma_{tool} \cup \Sigma_{mem} \cup \Sigma_{spawn} \cup \Sigma_{del}$, corresponding to message send/receive, tool invocation, memory read/write, sub-agent creation, and delegation-contract creation. A run is a finite or infinite trace $\rho = s_0 \to s_1 \to \cdots$, where transition k is labelled $a_k \in \Sigma$.

Each state contains the active contracts, budget ledger, and provenance links accumulated so far. We assume complete interposition: every authority-bearing delegation, tool action, and governed-memory write is mediated and logged. Consequently, each accepted action carries the active contract under which its agent operates and a provenance predecessor set identifying the ledger events, state versions, prior actions, and messages from which it causally derives.

3.2 Running example

We use a procurement scenario (ProcureBot) with one principal task: renew analytics software under a fixed budget, using vendor-side procurement data, while enforcing two procedural constraints: collect at least two quotes before payment and write an itemized record for each purchase. The active agents are a coordinator C, a quote-gathering research agent R, a runtime-spawned web agent W, and a purchase agent P. Tool surfaces are procurement db.read, email.send, browser.get, payments.execute, memory.read/write, and records.write; data labels include VENDOR, CONTRACT TERMS, EMPLOYEE PII, and PUBLIC.

The root contract (principal → C) grants bounded authority (including a global USD budget), data scope over vendor and contract labels, prohibitions on external disclosure of employee PII and unapproved

vendors, and obligations for two-quote evidence plus itemized purchase records. Hop 1 (C → R) is a correct attenuation that removes payment authority and preserves applicable prohibitions and obligations. Hop 2 (R → W) exposes dynamic membership: if R tries to grant browser.get without holding it, delegation is rejected by authority monotonicity; if R holds a domain-scoped browser capability, the child grant is a proper attenuation to one vendor domain. Hop 3 (C → P) is the erosion point for text-only delegation: objective text tends to persist while budget and obligation constraints are omitted; under DARC, the inherited contract and gateway checks block over-budget or under-evidenced payment attempts.

We reuse this example in Sections 4–6 to ground each property in a concrete enforcement moment: delegation-time mediation, action-time gateway checks, and incident-time revocation and recovery.

3.3 Threat model

We consider six attacker capabilities. (i) A compromised or careless delegator emits under-specified or malicious handoffs. (ii) A compromised delegee attempts unauthorized tool actions, data flows, or further delegation. (iii) Inter-agent message manipulation alters delegation context or objective interpretation (Agent-in-the-Middle style) [28]. (iv) Shared-memory poisoning introduces false or adversarial state that later delegations consume. (v) Tool-channel poisoning returns malicious content from external systems (for example, prompt-injected web pages). (vi) Colluding agents coordinate to bypass local guardrails or launder authority [32].

The trusted computing base is explicit: the enforcement layer (mediator, gateway, governed memory, ledger, incident controller), its policy store, and authenticated channels among these components are trusted. We also assume complete interposition over delegation, governed-memory writes, and tool calls that can change system or external state. Detection signals (for example topology anomalies in the style of [38]) are treated as evidence inputs; they do not directly authorize revocation without incident-policy approval.

Out of scope are model-weight compromise, hardware/OS side channels, and attacks that execute entirely outside mediated interfaces. These assumptions bound our formal guarantees and are revisited in Section 8.

4 Delegation Contracts

4.1 Contract structure

Definition 2

(Delegation contract) A delegation from agent i to agent j creates a contract

$C_{i \to j} = \langle o, cap, data, obl, proh, budget, expiry, prov, rev \rangle, (2)$

where o is the delegated objective, cap the permitted capabilities and tools, data the permitted data scope, obl obligations, proh prohibitions, budget resource and risk budgets, expiry temporal validity, prov the provenance chain, and rev revocation conditions.

Four design principles fix the enforceable meaning of these fields. (P1) Enforceable primitives only. Every field a mediator or gateway checks mechanically is drawn from a closed vocabulary: tool and operation identifiers, resource patterns, data labels from a deployment-fixed set, numeric budgets, timestamps, and agent and contract identifiers. The objective o is informative and is never an input to a deterministic check. (P2) Semantic clauses are declared, not hidden. Each obligation and prohibition carries a discriminator check $\in$\{deterministic, semantic\}: a deterministic clause references primitives and a checkable predicate, whereas a semantic clause carries natural-language text and receives only best-effort monitoring, which we report separately and for which we claim no guarantee. This makes the deterministic/semantic boundary syntactic. (P3) Attenuation by construction. A child contract is not formed freely and then audited against its parent; its constructor takes a parent contract and a requested delta and returns either a valid attenuation or a rejection (Section 4.2). (P4) Contracts are immutable; state lives in the ledger. Budget consumption, obligation satisfaction, and revocation status are append-only ledger facts keyed by contract identifier rather than mutations of the contract, which is what makes provenance and rollback tractable.

Concretely, a capability is a tuple $\langle$tool, operations, resources, limits$\rangle$, and a resource pattern is exactly one of the universal pattern $*$, an exact string, or a prefix

pattern P$*$(a single trailing $*$), so matching and subsumption are decidable and glob wildcards are rejected at construction time. The data scope is an antichain of labels; budget carries independent numeric axes (for the running example, US dollars and a tool-call count); expiry is a timestamp bounded by the parent's; prov is the append-only root-to-parent contract chain of Definition 1; and rev pairs revocation conditions with any cited independent authorizations (Property 4). The prototype's full JSON schema is reproduced in Appendix B.

4.2 Safety properties

The contract enforces six safety properties across three enforcement moments: delegation time (mediator), action time (gateway), and incident time (controller). In their predicate form, Properties 1–5 and the ledgered fragment of Property 6 are deterministic and receive formal treatment (Section 6, Appendix A); clauses that turn on the meaning of natural-language objectives or obligations are semantic tier and receive empirical accuracy analysis only (Section 7), never equivalent guarantees.

Property 1

(Authority monotonicity) cap(Ci$\rightarrow$j) $\subseteq$cap(i): a delegee never receives more authority than its delegator holds. Resource identifiers use a decidable prefix grammar (exact, prefix, or universal patterns), so capability subsumption is exact rather than approximated. Attenuable authority is a classical idea: capabilities as unforgeable references narrowed on transfer [22, 23], and contextual caveats a holder

may add before passing a credential on [25]. Our extension is to the multi-agent setting: attenuation is checked at every delegation event between autonomous agents rather than once on a static reference; it is enforced by construction (principle P3: the constructor emits a valid attenuation or a rejection); and the same handoff carries obligations and prohibitions alongside permissions (Property 3 onward), which permission-only capability models do not. The mediator (Section 5.2) is the enforcement point, and the check is deterministic.

Property 2

(Data-scope preservation) data(Ci→j) ⊆data(i). The label model is a configurable partial order, but we claim only what the prototype implements: a flat, deployment-defined label set (in the running example, PUBLIC, VENDOR, CONTRACT TERMS, EMPLOYEE PII) with a separate classification of sinks (internal, approved-vendor domain, unapproved external), under subset checking; we do not claim support for arbitrary lattices. Preservation is enforced twice: as a subset check on data at delegation time, and as a label-flow check at the capability gateway when data reaches a sink (Section 5.4), so an individually permitted read can still be blocked at an external sink. Both checks are deterministic.

Property 3

(Obligation preservation) Every obligation of i that is applicable to Ci→j is inherited verbatim, replaced by an approved transformation, or provably inapplicable to the child. Applicability is decided by an obligation's scope class: an action-local obligation gates a capability footprint and may be dropped only when that footprint is disjoint from the child's capabilities—safe because authority only attenuates, so no descendant can reacquire the gated capability (Appendix A); a workflow-contributing obligation, whose outputs feed another agent's gated action (e.g. quote provenance feeding a payment gate), must be inherited or transformed and is never dropped; a global obligation always propagates unless explicitly discharged. Discharge is not free rewriting: a transformation must be registered,

administrator-authored, versioned, and approved for the parent obligation, its role restrictions and preconditions must hold, and its output must conform to the schema. An LLM may propose which registered transformation applies, but the mediator checks registration, approval, preconditions, and conformance deterministically, and every invocation is ledgered. This property is split across tiers: inheritance and discharge of predicate obligations (registered evaluators over ledger facts) is deterministic and covered by the theorem; inheritance of obligations expressed only in natural language is semantic tier and receives empirical analysis (Section 7).

Property 4

(Revocation closure) Revoking a contract revokes every contract in its subtree of the delegation forest and withdraws every capability, data-scope grant, and pending action whose authority derives from that subtree, unless a descendant cites an independent authorization. Independent authorization is defined strictly: a descendant is spared only if it cites an authorization that (i) was validated and recorded as a

ledger edge at delegation time, (ii) is held by the same agent, (iii) is active and outside the revoked subtree, and (iv) covers the descendant's effective authority. At revocation time, the effective authority of the spared contract must be covered in full by the combined effective authority of its cited authorization set, re-evaluated over capabilities, data scope, expiry, obligations, prohibitions, and budget axes. We claim resistance only to forged citations, same-subtree laundering, and unrelated-authority laundering under these checks. The traversal and its guarantee appear in Section 6.2 and Theorem 1; the property is deterministic given the complete delegation graph.

Property 5

(Provenance completeness) Under complete interposition, every mediated event writes a ledger record, so every consequential action is traceable to an initiating objective (the root contract), a delegation chain (the provenance chain of Definition 1), the active contract, the acting agent, and the messages, state versions, and tool calls in its provenance predecessor set. Traceability is continuous rather than reconstructed after the fact, and it is what makes transitive revocation and lineage-based recovery decidable; the ledger and governed-memory components that record it are described in Sections 5.5 and 5.6. The property is deterministic given complete interposition.

Property 6

(Recovery consistency) After recovery completes, no active capability or persistent state depends exclusively on a revoked delegation chain: capabilities granted by revoked contracts are withdrawn (Property 4), and a memory entry is quarantined when its lineage terminates only in revoked chains, while state with at least one independent clean derivation may survive under the applicable policy. This mirrors causal rollback in optimistic execution [27] and compensation of committed steps in sagas [26], but it must restore semantic state (shared memory and task progress), not only transactional state. Two quarantine policies realize this property differently: conservative lineage quarantine (implemented), in which any tainted input taints the derived entry and may therefore remove valid state; and minimal lineage quarantine (planned), which retains an entry when at least one independently sufficient clean derivation path exists and therefore requires explicit AND/OR semantics over derivation sets. We do not describe the planned policy as exact. The two policies should be compared empirically on quarantine precision and recall, valid-work preservation, and residual compromise. Recovery consistency never implies that an irreversible external action has been undone; actions are classified as reversible, compensable, irreversible, or unknown.

5 The DARC Architecture

The guarantees in this section presume complete interposition: every delegation, tool call, and governed-memory operation passes through the mediator, gateway, and governed memory (Fig. 1). A framework adapter realizes this by routing the host framework's designated delegation, tool, and memory interfaces through the DARC core; our LangGraph adapter (darc/adapters/langgraph.py) hands each node only contract-bound handles—a governed toolbelt, a governed shared-state channel, and a mediated spawn

operation—never a raw tool, the ledger, or the mediator, and re-checks the active contract on every call. Interposition is therefore relative to the framework's mediated interfaces: it holds over the delegation, tool, and memory paths the runtime routes, but it does not confine a node that leaves those interfaces, because in-process Python provides no object-capability confinement (a node could reach a wrapped tool's private callable by attribute access, or import the core directly). This boundary is exactly the trusted-computing-base assumption of the threat model (Section 3.3) and the no-out-of-band-capability-acquisition premise of Lemma 2; out-of-band access is out of scope and revisited in Section 8.

5.1 Contract compiler

In the prototype the "compiler" is a transformation registry together with the mediator's discharge checks, rather than a full policy-to-contract compiler: administrators register versioned, approved obligation transformations, and at delegation time the mediator admits a transformation only if it is registered, approved for the parent obligation, and yields a schema-conforming result, following the classical operationalisation-of-norms template [10]. Claim discipline: the contribution is controlled compilation into a closed enforceable vocabulary, not automated policy compilation. Transformations are administrator-authored, versioned, and registered; an LLM may select or propose applications of approved transformations, and the mediator deterministically checks registration, approval, preconditions, and schema conformance; every invocation is ledgered. DARC never trusts an LLM to invent enforceable policy semantics.

5.2 Delegation mediator

The mediator is the sole constructor of contracts, which makes authority monotonicity structural: create root mints a principal's root grant, and delegate takes a parent contract and a requested delta and returns either a registered, valid child or an AttenuationError carrying every violated rule. It intercepts every handoff and sub-agent creation, enforces attenuation rules 1–7 (Section 6.1), and thereby realizes Properties 1–3 at delegation time; each cited independent authorization is validated for effective-authority coverage and recorded as a ledger edge before the child is registered. The decision is deterministic.

5.3 Communication-edge monitor

Every mediated event is written to the provenance ledger keyed by acting contract, agent, and derivation, which binds actions to their originating contract and provenance chain (Property 5); topology-anomaly detectors in the style of [38] are consumed here as evidence for incident authorities (Section 5.6), never as revocation triggers. Semantic clauses (natural-language obligations and prohibitions) are handled by a separate best-effort monitor kept strictly outside deterministic enforcement (design principle P2); the prototype ships a null monitor that abstains, and any monitor's verdicts are reported separately and never claimed as guarantees, consistent with the monitorability limits under partial observability [12, 14].

5.4 Capability gateway

The gateway mediates every tool call against the caller's active contract and is fully deterministic. In order it checks: liveness (a revoked or expired contract is refused with RevokedError, which is Property 4's action-time face); capability match on the (tool, operation, resource) triple under the prefix grammar; per-call numeric limits; budget spend within the ledger's remaining on every axis; label-flow prohibitions at sinks (Property 2's action-time face); and deterministic obligation gates, evaluated by registered predicate evaluators over ledger facts (Property 3). On refusal it logs a denial and returns the reasons; on success it records the spend and an action event (Property 5). Semantic obligations are never checked here.

5.5 Governed memory layer

Shared memory is a governed store: a write requires the writer's contract to hold memory.write and to carry the entry's labels within its data scope, and every entry records its writer contract and derivation lineage; reads are scope-filtered to active entries whose labels lie within the reader's data scope. Unlike static-policy governed memory [50], write authority is scoped by the delegation contract, and the recorded lineage is what makes incident-time quarantine executable (Property 6, Section 6.3) rather than audit-only.

5.6 Incident controller

The incident controller performs containment and recovery. Revocation requires an authenticated decision from an incident authority satisfying the active policy—a trusted principal or a k-of-n quorum of registered signals (an oracle authority is used only in controlled experiments)—so raw detector signals cannot revoke. On an authenticated decision it revokes the suspect contract, traverses the delegation graph to revoke descendants transitively subject to the strict sparing rule, settles reserved budgets back to surviving parents, and quarantines contaminated memory by lineage (Section 6). Tool-action compensation and task reassignment are specified but left to prototype v2.

6 Enforcement and Recovery Algorithms

6.1 Contract checking at delegation time

Algorithm 1

is the mediator's delegation check. It applies attenuation rules 1–7 and, only if all pass, constructs the child, records its cited authorization edges, and reserves its budget against the parent. Because the child is constructed by this procedure rather than checked after the fact, capability monotonicity (Assumption A3) is structural. Writing |cap| and |obl| for the parent's capability and obligation counts and |auth| for cited authorizations, the check costs O(|cap_delta| × |cap(Cp)| + |obl(Cp)| + |auth| × k), where k bounds one effective-authority coverage check. It is linear in contract size and cited authorizations and independent of global policy count.

6.2 Transitive revocation

Algorithm 2

performs transitive revocation. Given an authenticated incident decision (Assumption A6), it walks the trigger contract's delegation subtree, revoking each node unless the strict sparing rule (Assumption A5) applies. A spared node and its descendants remain active under the independently validated authorization. The controller then settles budgets and invokes lineage-based quarantine (Algorithm 3). Each node is enqueued at most once, so the traversal costs $O(|\text{subtree}(C)| + \Sigma_c |\text{auth}(c)| \times k)$, where k bounds an effective-authority coverage check.

Theorem 1

(Revocation closure under complete mediation) Assume complete mediation and interposition, capability monotonicity at every delegation hop, and the strict independent-authorization sparing rule of Assumptions A1–A6. Let R be the set of contracts marked revoked by revoke(C, δ). Then: (i) no contract or governed state changes unless δ is an authenticated incident decision; (ii) a descendant d of C is revoked if and only if no node other than C on the unique path from C to d is independently authorized with respect to C; and (iii) every surviving contract within subtree(C) is either directly covered by an active authorization outside that subtree, held by the spared agent, or is a valid attenuation-descendant of such a spared contract. Consequently, no capability or ledgered state deriving exclusively from C survives. The resistance claim is limited to forged-citation, same-subtree, and unrelated-authorization laundering under the stated assumptions. The proof is given in Appendix A.

6.3 State recovery

Algorithm 3

Conservative lineage quarantine.

Input: the set R of revoked contracts and the governed-memory derivation graph. (1) Mark every active memory entry written under a contract in R as tainted. (2) Repeatedly mark an entry tainted when any provenance predecessor is tainted, continuing to a fixpoint. (3) Quarantine all tainted entries so that they are excluded from subsequent governed reads. (4) Record the quarantine decision and lineage evidence in the append-only ledger.

The implemented procedure is fail-closed: any tainted input taints the derived entry. For $|E|$ governed-memory entries and p fixpoint passes, its cost is $O(|E| \times p)$, with p bounded by the longest derivation chain. Lineage completeness depends on Assumption A2, under which every governed write and derivation edge is mediated and recorded. Algorithm 3 realizes the ledgered-state fragment of Property 6. Tool-action compensation and task reassignment remain planned extensions, so recovery consistency is not claimed for irreversible external effects.

7 Evaluation Methodology and Exploratory Pilot

This section separates three forms of evidence that should not be conflated: formal guarantees about the model, mechanized implementation-conformance results for deterministic invariants, and empirical measurements of behavior that is not guaranteed by construction. The conformance and adapter-bypass results in Section 7.5 are deterministic, reproducible checks of implementation correspondence to the specification. No provisional defense percentages, confidence intervals, regression estimates, or cost figures are reported; those values will be inserted only from provenance-verified runs after the preregistered validation gates are satisfied.

7.1 Research questions and scope

RQ1. In an exploratory pilot, how do automated-judge codings of constraint retention, weakening, omission, contradiction, and unauthorized expansion vary with recursive natural-language delegation depth?

RQ2. Do the implemented mediator, gateway, ledger, and incident controller conform to the deterministic properties proved in Section 4 and Appendix A when challenged with attempts to strip constraints, amplify authority, exceed budgets, extend expiry, break provenance, or launder authorization?

RQ3. In meaning-dependent and stateful scenarios where DARC can fail, how do DARC and detection-oriented baselines compare on memory poisoning, semantic obligation loss, contract-compilation error, false-positive quarantine, valid-work preservation, and residual compromise?

RQ4. What latency, token, model-cost, and benign-task utility overheads are introduced by full DARC enforcement?

7.2 Evaluation structure and unit of analysis

Implementation-conformance study. Deterministic attack families are used to test whether the implementation matches the proved model. Attempts to widen a capability, omit a parent limit, drop a parent-budgeted axis, extend expiry, alter provenance, or retain same-subtree authority after revocation are expected to be rejected by construction. Success on these cases is evidence of implementation conformance, not an independent empirical discovery or a population-level comparison.

Exploratory H1 pilot. Delegation depth is varied over 1, 2, 4, and 8. Each handoff is coded for the disposition of every applicable constraint. The current coding pipeline uses an automated judge, with blinded human double-coding reserved for validation. Exact task-pair, seed, completion, and applicable-constraint counts will be taken from the frozen run bundle rather than fixed in the manuscript in advance.

Empirical robustness study. The current mechanism demonstration contains one canonical authored scenario for each of five threat families: constraint stripping, privilege amplification, compromised delegator or delegee, persistent memory poisoning, and false or malicious revocation. Clean/adversarial twins, controlled parameter variations, and repeated seeds may produce multiple runs per family, but

those repetitions do not create independent scenario diversity. Accordingly, the study is a curated mechanism demonstration and not a population benchmark.

Baselines are a no-guardrail control, an independent per-agent input/output guardrail, a central LLM policy judge, and a topology-aware detection-and-pruning defense. DARC variants are treated as ablations rather than baselines: without recovery, without provenance, and without attenuation.

7.3 Metrics and denominator discipline

Each metric has one fixed denominator. Constraint retention is retained applicable constraints divided by applicable constraints. Unauthorized expansion is unauthorized grants divided by completed delegation events. Executable violation rate is unauthorized actions attempted or executed divided by action opportunities. Attack success is successful adversarial runs divided by adversarial runs. False interruption is interrupted benign runs divided by benign runs. Contaminated-state recall and quarantine precision are computed over state instances, while valid-work preservation is computed over clean state or completed clean work, as declared before execution.

Loss severity is reported as a distribution, not only as a mean: retained/no loss, minor procedural loss, material obligation or restriction loss, and critical authority, privacy, or budget loss. The report will include the number of severity-3 losses, the share of chains with at least one critical loss, and the constraint types most frequently affected.

7.4 Automated-judge pilot and validation gate

The exploratory pilot uses automated-judge codings to exercise the measurement pipeline. A preregistered qualification gate requires at least 80% agreement and Cohen's kappa of at least 0.70 against a blinded reference set. The qualification round did not meet that gate, and independent validation is incomplete. Consequently, automated-judge depth summaries are not treated as confirmatory evidence for H1, and no odds ratio, confidence interval, or p-value is reported in this manuscript version.

After validation, the H1 report will show exact retained/applicable counts at each depth, task-pair-cluster bootstrap confidence intervals, the full severity distribution, missing-run accounting, and sensitivity analyses that remove one task pair at a time. The preregistered depth decision rule will be applied only to human-validated or gate-qualified codings.

7.5 Deterministic implementation conformance

The reviewed freeze candidate passed 75 mechanized tests, and freeze-manifest verification reported no drift between the frozen artifacts and the analyzed bundle. The suite exercises the seven delegation-time attenuation rules; capability operation, resource, and numeric-limit coverage; data-scope and expiry containment; obligation and prohibition propagation; budget reservation and subtree conservation; provenance append-only behavior; gateway liveness, capability, budget, label-flow, and obligation checks; authenticated revocation and strict sparing; and conservative lineage quarantine. Regression cases

explicitly exercise the two omission-as-escalation defects: omission of a parent capability limit and omission of a parent-budgeted axis. These outcomes are evidence that the implementation corresponds to the deterministic specification, not an empirical comparison on an uncertain outcome.

Adapter interposition is tested against six evasion vectors: direct tool references, dynamically added tools, nested subgraphs, direct state mutation, retry paths, and exception handlers. In the frozen suite, none of the six could execute a governed action or acquire authority outside the contract-bound path. This result is scoped to the framework's mediated interfaces; it does not establish confinement against arbitrary out-of-band access in in-process Python.

7.6 Empirical robustness protocol

Comparative empirical claims are reserved for outcomes not guaranteed by construction: semantic obligation inheritance, purpose limitation, memory poisoning whose harmful meaning is not captured by labels or predicates, contract-compilation fidelity, quarantine precision, valid-work preservation, residual compromise, and operational overhead. For every percentage, the manuscript will display the exact numerator and denominator; zero-event cells will include an exact interval rather than only 0.0%.

Ablations will use the same paired instances and metric definitions as full DARC. Unauthorized expansion will retain the delegation-event denominator in every table. Recovery ablations will report both residual compromise and valid-work preservation so that a variant cannot appear favorable merely by avoiding quarantine.

7.7 Reproducibility and artifact plan

Before empirical claims are added, the experiment bundle will pin the model identifier and provider, temperature and decoding settings, framework and adapter commits, tool configuration, prompts, run date, seed policy, task-generator version, judge rubric, baseline prompts, and statistical scripts. Raw requests, responses, tool events, ledger events, and coding decisions will be retained behind the bundle-level provenance gate. The harness and supporting artifacts are prepared for release, but repository and archival identifiers will be added only when a public archive exists.

8 Discussion and Limitations

8.1 Formalization as engineering evidence

The strongest evidence presently available is the interaction between the formal model and the implementation. Proving that capability coverage composes transitively exposed a latent escalation defect: a child contract could omit a numeric limit imposed by its parent, pass attenuation, and then be interpreted as unbounded by the gateway. Capability coverage now requires every parent limit to reappear at an equal or smaller value.

Budget conservation exposed the dual defect. A child could omit a budget axis imposed by its parent and thereby become unbounded on that axis. Mediator rule 5 now requires every parent-budgeted axis to be carried within the parent's remaining amount, and Theorem 8 gives the subtree bound. These defects yield a transferable design principle: in attenuation-by-construction delegation, every optional keyed constraint must default to the most restrictive value. Omission must never mean "unbounded."

8.2 Complete interposition and the trusted computing base

DARC's guarantees hold only over mediated delegation, tool, and governed-memory interfaces. The LangGraph adapter routes those designated paths through contract-bound handles and rechecks the active contract on each call. The six-vector bypass matrix reported in Section 7.5 found no route through direct tool references, dynamic tools, nested subgraphs, direct state mutation, retry paths, or exception handlers that could execute a governed action or acquire authority outside that path. This does not provide object-capability confinement for arbitrary in-process Python. A node that reaches a wrapped callable through private attributes, imports the enforcement core directly, or obtains an out-of-band tool has left the mediated interface and is outside the guarantee.

The mediator, gateway, governed memory, ledger, incident controller, policy store, and authenticated channels form the trusted computing base. Complete mediation and faithful logging are deployment assumptions, not theorems. A compromised enforcement component voids the guarantees in the same way that a compromised reference monitor voids the guarantees of a conventional capability system. Cryptographic contract authentication would harden artifact integrity, but would not itself confine in-process calls.

8.3 Semantic, compilation, recovery, and scale limits

The deterministic/semantic boundary is a boundary of claims. Natural-language objectives, semantic obligations, purpose limitations, and meaning-bearing memory are handled by a best-effort monitor that currently abstains; no formal guarantee is claimed for them. A deployment that relies on semantic clauses inherits the monitor's fallibility.

Contract-compilation fidelity remains the central unproved step. The formal results guarantee enforcement of the contract as written, not that the contract faithfully captures a principal's intent. DARC narrows this gap by allowing an LLM to select only among administrator-authored, registered transformations, rather than trusting it to invent enforceable policy semantics, but mis-specification remains possible.

Recovery is currently guaranteed only for ledgered authority and governed memory. Conservative lineage quarantine may remove valid state when clean and tainted derivations are mixed, while irreversible external effects cannot be undone by revocation. Tool compensation, task reassignment, durable storage, retention and compaction, and authenticated event chains remain engineering work. Provenance and per-action logging also grow with run length and delegation breadth.

8.4 Empirical evidence, generalization, and artifact status

The system-evaluation component is a curated mechanism demonstration with one canonical authored scenario per threat family, expanded through controlled variants and repeated seeds. It therefore supports mechanism inspection and implementation debugging, not population-level generalization. Deterministic families primarily test conformance to the proved core; the scientifically uncertain comparisons are concentrated in semantic and stateful failures such as memory poisoning and quarantine trade-offs.

The H1 pilot currently depends on an automated judge whose qualification round did not satisfy the preregistered agreement gate. Independent double-coding is incomplete, so confirmatory depth claims are deferred. The measurement pipeline, task definitions, rubric, and analysis code are prepared for archival release, but the manuscript does not claim that a public repository or DOI already exists.

9 Conclusion

Delegation is the operation through which authority, obligations, data access, and risk move through an open multi-agent system. Treating that operation as plain text leaves safety-relevant qualifications to probabilistic paraphrase. DARC instead makes each authority-bearing handoff a machine-checkable contract and mediates delegation, tool use, and governed memory against it. Under complete mediation and interposition, the deterministic fragment supports precise claims about attenuation, data-scope preservation, budget and expiry bounds, provenance, and revocation closure, while meaning-dependent clauses remain explicitly outside those guarantees. The formal effort also produced an immediate engineering result: it exposed two omission-based escalation defects and motivated the rule that optional keyed constraints must default to the most restrictive interpretation. The remaining empirical work should focus not on re-demonstrating properties enforced by construction, but on semantic clauses, contract compilation, memory poisoning, recovery precision, utility, and operational cost. DARC therefore contributes a disciplined boundary between what can be proved, what can be tested for conformance, and what must still be measured.

Declarations

Funding: The author received no specific funding for this work.

Competing interests: The author declares no competing interests.

Ethics approval: Not applicable.

Data and code availability: The prototype, evaluation harness, frozen task definitions, analysis scripts, and supporting run logs are prepared for public archival release. Repository and archival identifiers will be added when the archive is available; no public repository or DOI is claimed in this version.

Author contributions: Vinay Bamil is the sole author and was responsible for conceptualization, methodology, software, formal analysis, investigation, and manuscript preparation and revision.

Use of large language models: Large-language-model-based tools assisted with prototype and evaluation-harness implementation, analysis and tooling code, and drafting and editing portions of the manuscript. All research decisions, verification, interpretation, and final content are the author's responsibility. Large language models did not meet authorship criteria and are not listed as authors.

Author Contribution

V.B. is the sole author and was responsible for conceptualization, methodology, software, formal analysis, investigation, and the writing and revision of the manuscript.

References

1. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Trans. Computers C. **–29**(12), 1104–1113 (1980). https://doi.org/10.1109/TC.1980.1675516
2. Castelfranchi, C., Falcone, R.: Towards a theory of delegation for agent-based systems. Robot. Auton. Syst. **24**(3–4), 141–157 (1998). https://doi.org/10.1016/S0921-8890(98)00028-1
3. Castelfranchi, C., Falcone, R.: Trust Theory: A Socio-Cognitive and Computational Model. Wiley Series in Agent Technology. John Wiley & Sons (2010). https://doi.org/10.1002/9780470519851
4. Norman, T.J., Reed, C.: A logic of delegation. Artif. Intell. **174**(1), 51–71 (2010). https://doi.org/10.1016/j.artint.2009.10.001
5. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artif. Intell. Law. **7**(1), 97–113 (1999). https://doi.org/10.1023/A:1008319631231
6. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), pp. 527–534 (2002). https://doi.org/10.1145/544862.544867
7. Dastani, M., van der Torre, L., Yorke-Smith, N.: Commitments and interaction norms in organisations. Auton. Agent. Multi-Agent Syst. **31**(2), 207–249 (2017). https://doi.org/10.1007/s10458-015-9321-5
8. Esteva, M., Rodríguez-Aguilar, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the formal specification of electronic institutions. In: Agent Mediated Electronic Commerce: The European AgentLink Perspective. Lecture Notes in Artificial Intelligence, vol. pp. 126–147. Springer (2001). (1991). https://doi.org/10.1007/3-540-44682-6_8
9. Esteva, M., Rosell, B., Rodriguez-Aguilar, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), pp. 236–243 (2004). https://dl.acm.org/doi/10.5555/1018409.1018747

10. Aldewereld, H., Dignum, F., Garc´ıa-Camino, A., Noriega, P., Rodr´ıguez-Aguilar, J.A., Sierra, C.: Operationalisation of norms for usage in electronic institutions. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 223–225 (2006). https://doi.org/10.1145/1160633.1160669
11. Dignum, V.: A model for organizational interaction: Based on agents, founded in logic. PhD thesis, Utrecht University SIKS Dissertation Series 2004-1. (2004). https://dspace.library.uu.nl/handle/1874/890
12. Bulling, N., Dastani, M., Knobbout, M.: Monitoring norm violations in multi-agent systems. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), pp. 491–498 (2013). https://dl.acm.org/doi/10.5555/2484920.2484999
13. Alechina, N., Bulling, N., Dastani, M., Logan, B.: Practical runtime norm enforcement with bounded lookahead. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), pp. 443–451 (2015). https://www.ifaamas.org/Proceedings/aamas2015/aamas/p443.pdf
14. Criado, N., Such, J.M.: Norm monitoring under partial action observability. IEEE Trans. Cybernetics. **47**(2), 270–282 (2017). arXiv:1505.03996
15. Criado, N.: Resource-bounded norm monitoring in multi-agent systems. J. Artif. Intell. Res. **62**, 153–192 (2018). https://doi.org/10.1613/jair.1.11206
16. Alechina, N., Halpern, J.Y., Kash, I.A., Logan, B.: Decentralised norm monitoring in open multi-agent systems. In: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016) Extended abstract; arXiv:1602.06731. https://arxiv.org/abs/1602.06731 (2016)
17. Alechina, N., Halpern, J.Y., Kash, I.A., Logan, B.: Incentivising monitoring in open normative systems. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, pp. 305–311 (2017)
18. Tzeng, S.-T., Ajmeri, N., Singh, M.P.: Norm enforcement with a soft touch: Faster emergence, happier agents. In: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024) (2024). https://dl.acm.org/doi/10.5555/3635637.3663046
19. Dastani, M., Sardiña, S., Yazdanpanah, V.: Norm enforcement as supervisory control. In: PRIMA 2017: Principles and Practice of Multi-Agent Systems. Springer (2017). https://doi.org/10.1007/978-3-319-69131-2_20
20. Sarkar, A., Blair, C., Muresanu, A.I., et al.: Normative Modules: A Generative Agent Architecture for Learning Norms that Supports Multi-Agent Cooperation. arXiv:2405.19328 (2024). https://arxiv.org/abs/2405.19328
21. He, S., Ranathunga, S., Cranefield, S., Savarimuthu, B.T.R.: Norm violation detection in multi-agent systems using large language models: A pilot study. In: Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XVII (COINE 2024). Lecture Notes in Artificial Intelligence, vol. 15398, pp. 146–159. Springer (2025). https://doi.org/10.1007/978-3-031-82039-7_10

22. Dennis, J.B., Van Horn, E.C.: Programming semantics for multiprogrammed computations. Commun. ACM. **9**(3), 143–155 (1966). https://doi.org/10.1145/365230.365252
23. Miller, M.S.: Robust composition: Towards a unified approach to access control and concurrency control. PhD thesis, Johns Hopkins University (2006). http://www.erights.org/talks/thesis/
24. Ellison, C.M., Frantz, B., Lampson, B., Rivest, R.L., Thomas, B.M., Ylonen, T.: SPKI certificate theory. Technical Report RFC 2693, Internet Engineering Task Force (1999). https://doi.org/10.17487/RFC2693
25. Birgisson, A., Politz, J.G., Erlingsson, U., Taly, A., Vrable, M., Lentczner, M.: Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In: Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS 2014) (2014). https://doi.org/10.14722/ndss.2014. 23212
26. Garcia-Molina, H., Salem, K.: Sagas. In: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pp. 249–259 (1987). https://doi.org/10.1145/38713.38742
27. Jefferson, D.R.: Virtual time. ACM Trans. Program. Lang. Syst. **7**(3), 404–425 (1985). https://doi.org/10.1145/3916.3988
28. He, P., Lin, Y., Dong, S., et al.: Red-teaming LLM multi-agent systems via communication attacks. In: Findings of the Association for Computational Linguistics: ACL 2025 (2025). arXiv:2502.14847. https://arxiv.org/abs/2502.14847
29. Liang, R., Yin, L., Chen, J., et al.: Tipping the Dominos: Topology-Aware Multi-Hop Attacks on LLM-Based Multi-Agent Systems. arXiv:2512.04129 (2025). https://arxiv.org/abs/2512.04129
30. Peign´e, P., Kniejski, M., Sondej, F., et al.: Multi-agent security tax: Trading off security and collaboration capabilities in multi-agent systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39 (2025). https://doi.org/10.1609/aaai.v39i26.34970
31. Zha, M., Wang, X.: Autonomous LLM Agent Worms: Cross-Platform Propagation, Automated Discovery and Temporal Re-Entry Defense. arXiv:2605.02812 (2026). https://arxiv.org/abs/2605.02812
32. Zeng, X., MacArthur, R., Kryven, M., et al.: I Know It's Unfair, Do It Anyway: LLM Agents Exploiting Explicitly Unfair Tools for Voluntary Collusion in Strategic Games. OpenReview, ICLR 2026 submission. forum?id=5N2CafbMbJ (2026). https://openreview.net/
33. Hammond, L., Chan, A., Clifton, J., et al.: Multi-agent risks from advanced AI. Technical Report Technical Report #1, Cooperative AI Foundation arXiv:2502.14143. (2025). https://arxiv.org/abs/2502.14143
34. Witt, C., Krawiecka, K., Krawczuk, I., et al.: Open Challenges in Multi-Agent Security: Towards Secure Systems of Interacting AI Agents. arXiv:2505.02077 (2025). https://arxiv.org/abs/2505.02077
35. Wang, H., Poskitt, C.M., Sun, J.: Agentspec: Customizable runtime enforcement for safe and reliable LLM agents. In: Proceedings of the 48th International Conference on Software Engineering (ICSE 2026) (2026). https://cposkitt.github.io/files/publications/agentspecllmenforcementicse26.pdf

36. Luo, W., Dai, S., Liu, X., Banerjee, S., Sun, H., Chen, M., Xiao, C.: AGrail: A lifelong agent guardrail with effective and adaptive safety detection. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025) (2025). https://aclanthology.org/2025.acl-long.399.pdf
37. Yu, M., Wang, S., Zhang, G., et al.: Netsafe: Exploring the topological safety of multi-agent system. In: Findings of the Association for Computational Linguistics: ACL 2025 (2025). arXiv:2410.15686. https://arxiv.org/abs/2410.15686
38. Wang, S., Zhang, G., Yu, M., et al.: G-safeguard: A topology-guided security lens and treatment on LLM-based multi-agent systems. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025) arXiv:2502.11127, code: (2025). https://github.com/wslong20/Gsafeguard. https://arxiv.org/abs/2502.11127
39. Zhu, Y., Zhang, C., Shi, X., et al.: MASTER: Multi-Agent Security Through Exploration of Roles and Topological Structures – A Comprehensive Framework. arXiv:2505.18572 (2025). https://arxiv.org/abs/2505.18572
40. Lee, H., Yun, V.-D., Oh, H., et al.: Robust Multi-Agent LLMs under Byzantine Faults. arXiv:2605.09076 (2026). https://arxiv.org/abs/2605.09076
41. Ebrahimi, S., Dehghankar, M., Asudeh, A.: An Adversary-Resistant Multi-Agent LLM System via Credibility Scoring. arXiv:2505.24239 (2025). https://arxiv.org/abs/2505.24239
42. Feng, Y., Pan, X.: SentinelNet: Safeguarding Multi-Agent Collaboration Through Credit-Based Dynamic Threat Detection. arXiv:2510.16219 (2025). https://arxiv.org/abs/2510.16219
43. Ma, Y., Yang, Z., Zhang, Y.: Catching the Infection Before It Spreads: Foresight-Guided Defense in Multi-Agent Systems. arXiv:2605.01758 (2026). https://arxiv.org/abs/2605.01758
44. Siu, V., He, J., Montgomery, K., et al.: A Framework for Formalizing LLM Agent Security. arXiv:2603.19469 (2026). https://arxiv.org/abs/2603.19469
45. Venkataramani, V., Shi, H., Ke, Z., et al.: MAS-ProVe: Understanding the Process Verification of Multi-Agent Systems. arXiv:2602.03053 (2026). https://arxiv.org/abs/2602.03053
46. ElSayed-Aly, I., et al.: Safe multi-agent reinforcement learning via shielding. In: Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), pp. 483–491 (2021). https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p483.pdf
47. Xiao, W., Lyu, Y., Dolan, J.: Model-based dynamic shielding for safe and efficient multi-agent reinforcement learning. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023) arXiv:2304.06281. https://arxiv.org/abs/2304.06281 (2023)
48. Brorholt, A.H., Larsen, K.G., et al.: Compositional shielding and reinforcement learning for multi-agent systems. In: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025) arXiv:2410.10460. (2025). https://dl.acm.org/doi/10.5555/3709347.3743554

49. Kouvaros, P., Lomuscio, A., Pirovano, E., Punchihewa, H.: Formal verification of open multi-agent systems. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), pp. 179–187 (2019). https://dl.acm.org/doi/10.5555/3306127.3331691
50. Margalit, Y., Cohen-Inger, N., Avram, E., et al.: Governed Shared Memory for Multi-Agent LLM Systems. arXiv:2606.24535. System name: MemClaw (2026). https://arxiv.org/abs/2606.24535
51. Li, C., Zhang, L., Zhai, J., et al.: Towards Security-Auditable LLM Agents: A Unified Graph Representation. arXiv:2605.06812. Framework name: Agent-BOM (2026). https://arxiv.org/abs/2605.06812
52. Kavathekar, I., Jain, H., Rathod, A., et al.: TAMAS: Benchmarking Adversarial Risks in Multi-Agent LLM Systems. arXiv:2511.05269 (2025). https://arxiv.org/abs/2511.05269
53. Le Mercier, A., Develder, C., Demeester, T.: GAMBIT: A Three-Mode Benchmark for Adversarial Robustness in Multi-Agent LLM Collectives. arXiv:2605.09027 (2026). https://arxiv.org/abs/2605.09027
54. Rose, A., Cullen, C., Abdelnabi, S., et al.: Detecting Multi-Agent Collusion Through Multi-Agent Interpretability. arXiv:2604.01151. Introduces the NAR-CBench benchmark (2026). https://arxiv.org/abs/2604.01151

Figures

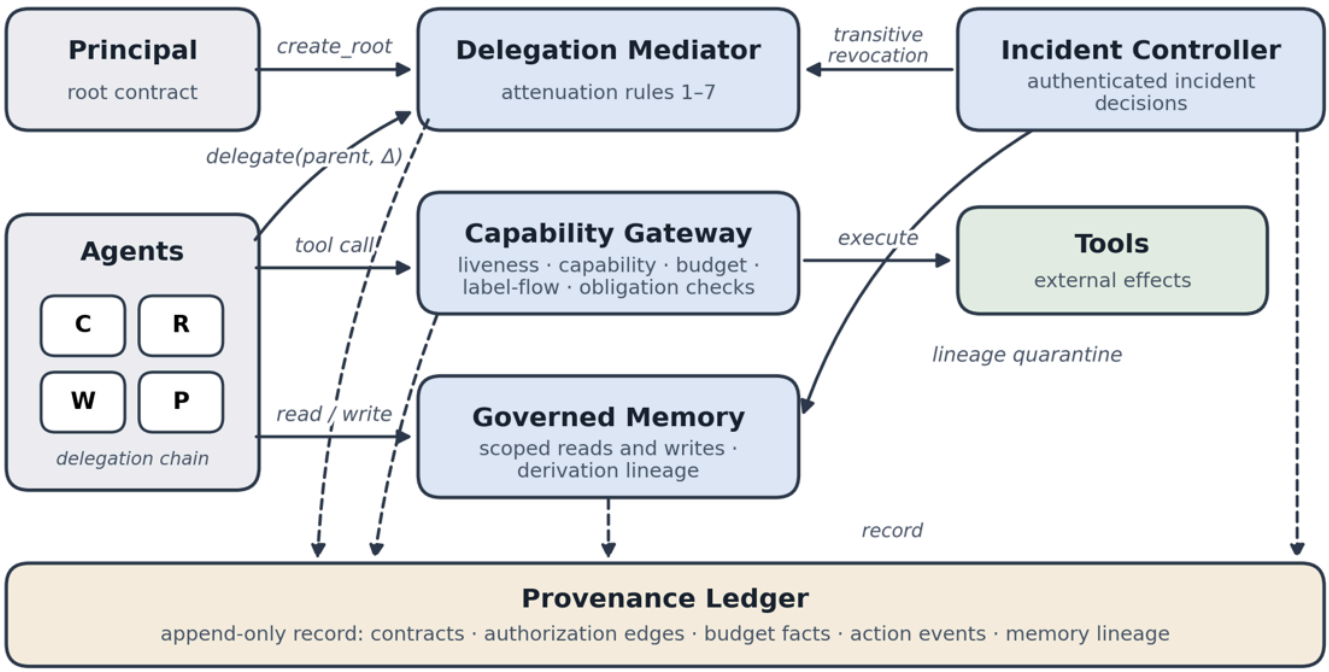


Figure 1

The DARC enforcement architecture. Agents delegate through the mediator, act through the capability gateway, and read and write through governed memory. Every component records to the append-only provenance ledger; authenticated incident decisions trigger transitive revocation and lineage-based quarantine