TRUSTWORTHY MULTI-AGENT SYSTEMS: MITIGATING SEMANTIC DRIFT WITH THE ARGENT SIGNALING PROTOCOL

XTAI 2026, AISB CONVENTION, UNIVERSITY OF SUSSEX, UK

Anantha Sharma
Synechron Inc
Charlotte, NC, USA

July 2026

ABSTRACT

When multi-agent LLM systems produce bad answers, not all failures are equal: some answers are grounded in the right material but incomplete, while others are simply ungrounded and should be stopped. Current retry strategies treat both cases identically, leaving human supervisors unable to tell whether a retry was warranted or whether the system should have halted.

We introduce the *Argent* Signaling Protocol (ASP) (*argent* as in silver, signaling clarity and audit-grade transparency; distinct from Answer Set Programming). ASP is a compact machine-readable header that accompanies every AI-generated response with structured quality signals: certainty (@C), grounding (@G), stochasticity (@S), and an assumption index that classifies the evidentiary basis of each claim. These signals let a controller distinguish *repairable* failures from *containment* failures and route each case differently.

We evaluate ASP in two complementary settings: a standalone controller for document-grounded QA and a sidecar-gated two-agent pipeline. Standalone, ASP raises aggregate passes from 12/81 to 21/81 on the 27-question Array BioPharma/Ono benchmark (10 recoveries, 1 regression; aggregate one-sided McNemar $p = 0.006$) and from 25/429 to 34/429 on a 19-contract CUAD breadth extension (9 recoveries, 0 regressions; $p = 0.002$). On heterogeneous CUAD the utility gain is modest (+2.1 absolute points aggregate); the controller's value shifts from repair to containment, with the safety profile preserved. A single-model multi-agent case study (SmolLM3, 27 questions, 14 ungrounded baseline outputs) shows the ASP sidecar blocking 24/27 outputs and preventing every ungrounded one from reaching the downstream decision agent, controlling for semantic drift before it propagates across agent boundaries.

Keywords Multi-Agent Systems · Explainable AI · Semantic Drift · Uncertainty · Grounded Generation · Large Language Models

[1]

1 Introduction

As Large Language Models (LLMs) move from isolated prompting to orchestrated Multi-Agent Systems (MAS), intermediate outputs increasingly function as operational state rather than disposable prose [1, 2]. In *document-grounded* settings, downstream agents may act on upstream answers that are only partially supported, poorly cited, or already drifting away from the source text. The problem is not only whether an answer is wrong, but whether the system can tell what kind of wrong it is before that answer is reused.

[1] An extended version of this paper including the full protocol specification (the stochasticity-modulated centroid update, adaptive assumption-decay mechanism, and the full per-signal weight rationale), all four trajectory cards, and detailed per-model analyses will be made available on arXiv.

Consider an AI system answering legal questions over a pharmaceutical license agreement. One answer cites the correct section ("Section 1.6, Change in Control") and recovers some of the required terms, but misses branches about mergers and asset sales. Another produces fluent legal-sounding prose with no citations and no grounding. These failures are operationally different. The first is *repairable*: the right material was retrieved, and a second attempt may recover the missing branches. The second requires *containment*: retrying an answer that was never grounded is more likely to polish fabrication than fix it. Most retry strategies, however, treat both failure modes the same.

The accumulation of untracked assumptions is *semantic drift*: locally plausible steps compound into globally invalid trajectories because downstream agents consume upstream assertions as if they were settled facts. A familiar human analogue is the children's game of "telephone" (sometimes called "Chinese whispers"), where each retelling can introduce small distortions that accumulate into a final message far from the original. This problem is related to calibration and uncertainty in neural models [3, 4], but multi-agent systems reproduce the same dynamic at much higher speed and scale by recursively reusing model outputs as fresh context, allowing drift to propagate across a pipeline before a supervisor can intervene. The result is not only incorrect final answers, but also a loss of auditability about whether the system is repairing grounded partial answers, refining unsupported ones, or silently giving up. Explainable and risk-managed deployments therefore require message-level traceability, interpretable failure state, and controls that are measurable rather than purely rhetorical [5, 6]. ASP extends the documentation logic behind Datasheets for Datasets [7] and FactSheets [8] from datasets and services to individual agent-to-agent messages.

This gap can be addressed using the Argent Signaling Protocol (ASP), a lightweight protocol that attaches structured quality signals to every AI-generated response. ASP builds on the Argent Framework [9], which introduced governance-first primitives for composing, orchestrating, and governing enterprise AI agents. Where the Argent Framework defines how agents are built and bounded, ASP defines how they communicate grounding, certainty, stochasticity, and assumptions, so a governing controller can distinguish repairable outputs from containable ones instead of relying on opaque retry loops.

Contributions

This paper makes three contributions:

- Define ASP and its sidecar deployment model.
- Evaluate ASP as a standalone controller on a 27-question grounded-QA benchmark and a 19-contract / 143-question CUAD breadth extension, where aggregate passes rise from 12/81 to 21/81 on the deep dive (10 fail-to-pass recoveries, 1 regression) and from 25/429 to 34/429 on the breadth test (9 fail-to-pass, 0 regressions).
- Present a single-model case study of multi-agent governance in which the sidecar blocks every ungrounded upstream output from reaching a downstream decision agent on the 27-question benchmark.

2 Related Work

Agent frameworks CAMEL [2] and AutoGen [1] show that multi-agent decompositions can attack problems a single LLM call cannot. Both exchange free-form language or task-specific tool calls; neither carries a standardized uncertainty or provenance contract. When Agent A tells Agent B "the contract defines Change in Control in Section 1.6," Agent B cannot tell whether Agent A was quoting retrieved evidence or generating from memory. The Argent Framework [9] addresses agent composition and budget-bounded execution at the orchestration level; ASP occupies the protocol layer between the agent and its governor.

Self-correction Self-Refine [10] and Reflexion [11] demonstrate that feedback-driven retries can improve answers. In high-stakes settings the feedback stays rhetorical: after three retries an auditor sees that the system tried three times but cannot tell whether any attempt was grounded, whether confidence rose or fell across attempts, or whether halting earlier would have been safer. ASP makes those signals explicit and machine-readable, so every retry carries its own quality report card.

Calibration and uncertainty Modern neural networks are often miscalibrated [3]; LLMs sometimes know what they know under the right elicitation protocol [4]; and verbalized confidence can diverge from true correctness [12, 13]. ASP does not claim access to true posterior confidence. It operates on auditable proxies of grounding and instability and asks whether those proxies are sufficient, in practice, to route between repair and containment.

Documentation and reporting standards Datasheets for Datasets [7] and FactSheets [8] standardize documentation for training data and AI services respectively. ASP extends that practice to the inter-agent communication layer: each message carries its own provenance, confidence, and assumption metadata, so the entire multi-agent conversation is auditable as a chain of documented decisions.

Table 1 compares ASP to the closest related systems along four axes. To our knowledge ASP is the first agent protocol that exposes all four at the message boundary.

3 The Argent Signaling Protocol

The core idea is simple: before accepting any AI-generated answer, attach a machine-readable "report card" that rates the answer on multiple dimensions. A supervisor (human or automated) can read the card at a glance to decide whether to trust the answer, ask for a revision, or discard it. As in clinical lab reports, the header carries quality flags alongside the result. At each step the controller scores the current draft against a fixed ideal QA reference profile (§4.2) and updates per-agent error counters; ASP exposes the minimal structured state needed to update these quantities deterministically, so every routing decision can be reproduced from the telemetry log.

Protocol scope vs. benchmark scope ASP is defined at the protocol level, and this paper evaluates it across both standalone and multi-agent settings. The benchmarks in Sections 5–7 exercise the header format (§3.1), the signal fields (§3.2), the assumption index (§3.4), the drift monitor (§4.2), and the routing logic (§4.3). The same protocol is studied in two operational regimes: bounded controller loops for document-grounded QA and sidecar-mediated agent-to-agent traffic, letting us evaluate ASP both as a per-answer control surface and as an inter-agent governance layer.

3.1 Header Format

Every ASP-instrumented response carries a compact header alongside its natural-language content:

`@C`	`@G`	`@S`	`A` (Assumption Index)
`X`	`Y`	`Z`	`[K:id, L:id, P:id, H:id]`

On the wire, the same header is serialized as `[@C:X; @G:Y; @S:Z; A:[K:id, L:id, P:id, H:id]]`. Each signal field (@C, @G, @S) is a hexadecimal digit from 0 (lowest) to F (highest, i.e. 15), representing a quality score on a 16-point scale. We use 16 levels so each signal fits in a single hexadecimal digit, preserving a compact fixed-width header; the controller itself operates on the underlying continuous values, so the serialization choice does not constrain the routing logic. The `A:[...]` field is the assumption index. A well-grounded answer that cites the right source might carry `[@C:D; @G:F; @S:2; A:[K:42, L:09]]`: certainty high, grounding maximal, stochasticity low, one known fact and one learned derivation. Contrast `[@C:3; @G:1; @S:C; A:[H:17]]`: certainty very low, grounding nearly absent, stochasticity high, only a hypothetical assumption. The controller does not need to read the answer text to know this response should not be passed downstream. The header is compact, auditable (every field mechanically derivable, no opaque learned components), and protocol-level (it travels with the response through any downstream agent). We prioritize interpretability over optimality: a controller that makes the right decision 85% of the time and can explain each one is more useful in regulated settings than one that makes the right decision 90% of the time opaquely.

3.2 Signal Fields

The three signal fields capture complementary dimensions of response quality. **@C (Certainty)** is a controller-visible certainty proxy for answer reliability; in the benchmark, when no separate confidence channel is available, Equation 3 derives it from the grounding and stochasticity proxies, so it should not be read as an empirically calibrated probability. **@G (Grounding)** reports how well the answer is anchored in retrieved evidence; @G of F means nearly every word in the answer appears in the source chunks, @G of 0 means the answer shares almost no vocabulary with the source. **@S (Stochasticity)** reports how much of the answer is unsupported or unstable; a high @S is a warning sign that the model is generating material that cannot be verified against the source.

3.3 Estimating ASP Fields from Closed-API Agents

ASP specifies the signal contract, not the estimator. Candidate black-box estimators include verbalized confidence for @C [13, 14], NLI evidence coverage for @G, and k-sample disagreement for @S [12] — none requires logit access. We outline this family to indicate the intended deployment surface; we do *not* validate it here. Every result in this paper

Table 1: ASP versus closely related multi-agent and self-correction systems. ✓ = present at the protocol or orchestration level; × = absent; partial = available as natural-language critique without machine-readable fields.

System	Structured uncertainty	Provenance / assumption channel	Drift monitor	Auditable circuit-break
AutoGen [1]	×	×	×	×
CAMEL [2]	×	×	×	×
Self-Refine [10]	partial	×	×	×
Reflexion [11]	partial	×	×	×
ASP (this work)	✓ (@C, @S)	✓ (@G, A:[K,L,P,H])	✓ (JSD, τ)	✓ (Eq. 6)

uses the white-box token-overlap estimator of Section 3.2, chosen for transparency. The closed-API path is therefore proposed rather than demonstrated, and its calibration against the white-box estimator is future work.

3.4 Assumption Index

Beyond the three numeric signals, ASP provides a structured *assumption index* that classifies the evidentiary basis of each claim. Every assumption referenced in the answer is tagged with one of four categories (Table 2). A response built on K (known) and L (learned) assumptions is citing and interpreting the actual document; a response built on H (hypothetical) is speculating. The controller can decide: repair when the answer is mostly K/L with missing terms; containment when it is mostly H with no grounding.

Table 2: Assumption categories in the ASP header

Tag	Category	Meaning
K	Known	Established fact directly stated in the source document.
L	Learned	Derived during processing (e.g., identifying a chunk with the definition).
P	Projected	Inferred from partial evidence; the source suggests but does not state it.
H	Hypothetical	Speculative; no direct evidence; the model is reasoning beyond the source.

3.5 Sidecar Deployment Model

ASP is an interception layer between agents rather than inside them, analogous to service-mesh sidecars. Figure 1 shows the architecture: a single shared bus with one tap per agent. An agent's only outbound channel is its tap; there is no direct agent-to-agent path. The bus computes @C/@G/@S, drift, and the combined assumption index, and is the sole enforcement surface for inter-agent traffic. Agents can be swapped without changing the bus, and vice versa.

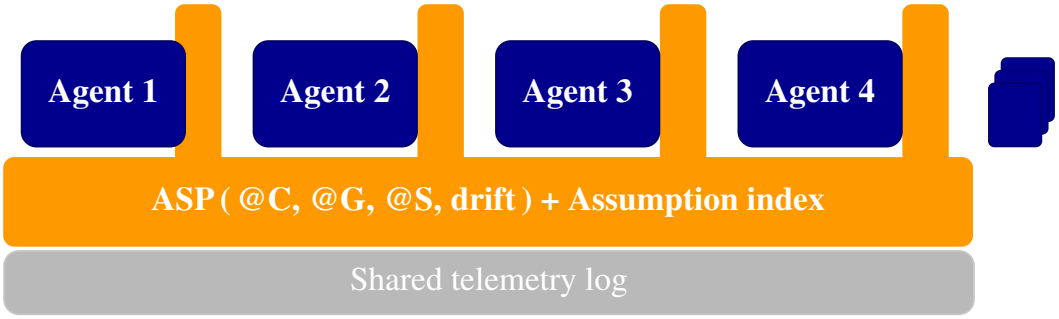

Figure 1: Sidecar-mediated architecture. The ASP layer (orange) is a shared bus that all agents tap into via dedicated sidecars; agents have no direct path to one another, so every inter-agent message is forced through the bus, which acts as the sole enforcement surface.

Figure 2 shows the operational pipeline: Agent A's output passes through ASP Sidecar 1 (computes @C/@G/@S and drift, makes a deterministic gate decision); Agent B's output is similarly gated. Sidecar-triggered regenerations are bounded by the error budget ($E_{\max} = 2$) and a flow-level TTL; exhausted items go to a human review queue. Every

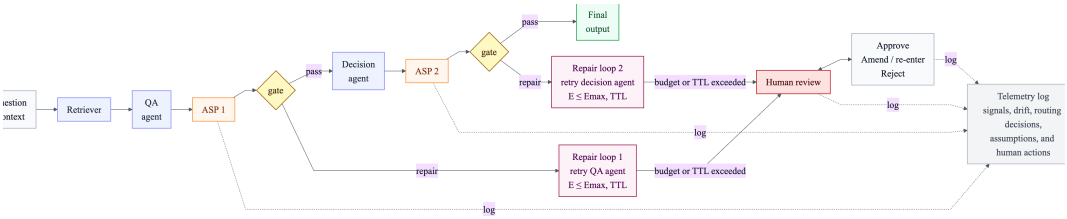

Figure 2: Operational view of the same architecture as Figure 1, per agent boundary. "ASP Sidecar 1/2" are not separate buses — each is one tap into the shared bus of Figure 1, shown twice because each boundary is gated independently. Failed items enter bounded repair loops; unresolved cases escalate to human review; telemetry records every decision.

sidecar decision, repair attempt, and human action is logged. ASP operates in two modes within the same deployment: standalone (single agent with a controller loop, §6) and multi-agent (sidecars at every boundary, §7).

4 Controller Implementation

This section describes how ASP signals are computed and used in the benchmark controller. The protocol (§3) defines *what* signals are carried; this section defines *how* they are derived and *how* the controller acts on them. The same protocol could be implemented with different estimators; the benchmark uses a token-overlap estimator because it is transparent, reproducible, and sufficient for document-grounded QA.

4.1 Signal Computation

For each generated draft, three raw features are extracted by comparing the answer tokens against the source material:

- o_c (*context overlap*): fraction of answer tokens appearing in any retrieved chunk.
- s_{cite} (*citation score*): 1.0 for a valid retrieved-chunk citation, 0.4 for a citation not in the retrieved set, 0.0 otherwise.
- n (*novel_ratio*): fraction of answer tokens found in neither the retrieved context nor the question (excludes question-echo tokens).

A fourth feature, o_q (*question overlap*), feeds the drift vector but not the signal fields directly. Grounding and stochasticity are then:

$$@G = \text{clamp}\big(0.60 \cdot o_c + 0.40 \cdot s_{\text{cite}}\big) \tag{1}$$

$$@S = \text{clamp}\big(0.70 \cdot n + 0.30 \cdot (1 - s_{\text{cite}})\big) \tag{2}$$

If the agent exposes an explicit confidence signal $c_{\text{conf}} \in [0, 1]$, the controller sets $@C = \text{clamp}(c_{\text{conf}})$. When no such signal is available, it falls back to

$$@C = @G \cdot (1 - @S) \tag{3}$$

where $\text{clamp}(x) = \min(1, \max(0, x))$. Grounding is weighted toward o_c because using the source document's language is the strongest evidence of document awareness; stochasticity is weighted toward n because fabricated material is the primary risk in grounded QA; the fallback certainty rule keeps @C low whenever grounding is weak or novelty is high. In the benchmark reported here, the local models did not emit a separate confidence channel, so Equation 3 supplied @C. Each value is compressed into a single hexadecimal digit for the ASP header; the controller operates on the raw floats.

4.2 Drift Monitoring

Individual signal values describe a single draft but do not reveal a trend. Drift monitoring measures how far each answer's quality profile sits from what a well-grounded response typically looks like. Jensen–Shannon divergence (JSD) [15] is used as a bounded, symmetric drift score on the L1-normalized feature vector. This is a computational choice, not an information-geometric claim: $\mathbf{v}$ is a heterogeneous quality-feature vector, not a parametric distribution over a task sample space. With base-2 logarithms, $\text{JSD}(P\|Q) \in [0, 1]$, and $\sqrt{\text{JSD}}$ is a proper metric on normalized vectors [16].

Each draft produces a four-element quality-profile vector. Components are floored at 0.01 and then apply L1 normalization:

$$\tilde{\mathbf{v}} = \big(\max(0.01, o_c),\ \max(0.01, s_{\text{cite}}),\ \max(0.01, n),\ \max(0.01, o_q)\big), \qquad \mathbf{v} = \frac{\tilde{\mathbf{v}}}{\|\tilde{\mathbf{v}}\|_1}. \tag{4}$$

Drift is the square root of the JSD between $\mathbf{v}$ and a fixed ideal QA reference $\mathbf{v}^* = (0.45, 0.35, 0.03, 0.17)$ (high context overlap, strong citation, low novel content, moderate question overlap):

$$d = \sqrt{\mathrm{JSD}(\mathbf{v} \,\|\, \mathbf{v}^*)} = \sqrt{\tfrac{1}{2}\big[\mathrm{KL}(\mathbf{v}\|\mathbf{m}) + \mathrm{KL}(\mathbf{v}^*\|\mathbf{m})\big]}, \quad \mathbf{m} = \tfrac{\mathbf{v}+\mathbf{v}^*}{2} \tag{5}$$

In the benchmark, a Qwen answer that cites the right chunk and uses the source language produces $d \approx 0.11$; a Dobby answer that cites correctly but misses required terms produces $d \approx 0.20$; a SmolLM3 answer that paraphrases without citations produces $d \approx 0.62$. Because JSD is bounded and symmetric, 0 means identical to the ideal and 1 means maximally different.

Sensitivity to miscalibrated confidence signals A reasonable concern is what happens when the elicited certainty signal is itself miscalibrated. The routing rule (Eq. 6) consumes raw grounding g, stochasticity s, and drift d; the certainty field @C is reported in the header for downstream supervisors but is not a routing input, and the drift computation (Eqs. 4–5) operates over independent token-level features (o_c, s_{cite}, n, o_q) — @C is absent from $\mathbf{v}$. Verbal-confidence bias of the form $@C_{\text{obs}} = @C_{\text{true}} + \varepsilon$ therefore does not enter the drift threshold check at all, and per-agent bias in @C would be detectable through cross-agent disagreement in a multi-agent deployment. The relevant residual sensitivity is to noise in the token-level features themselves (most acutely n, the novel-token ratio, which is a small set-membership count): characterising $\partial d/\partial \mathbf{v}$ near the threshold and translating estimator noise on each component into a flip rate requires an ablation we do not perform here; it is a target for the follow-on sensitivity work noted in Section 4.4.

4.3 Routing and Failure Classification

Given the signal values and drift score, the controller applies a deterministic routing policy as a priority cascade, with the first satisfied clause taking precedence:

$$R(\text{agent}) = \begin{cases} \text{Yield}, & \text{if } E \geq E_{\max} \\ \text{Regenerate}, & \text{if } E < E_{\max} \wedge (g < g_{\min} \vee s > \sigma_{\max} \vee d > \tau_{\text{crit}}) \\ \text{Warn}, & \text{if } E < E_{\max} \wedge g \geq g_{\min} \wedge s \leq \sigma_{\max} \wedge \tau_{\text{warn}} < d \leq \tau_{\text{crit}} \\ \text{Execute}, & \text{if } E < E_{\max} \wedge g \geq g_{\min} \wedge s \leq \sigma_{\max} \wedge d \leq \tau_{\text{warn}}. \end{cases} \tag{6}$$

Here g and s are the raw continuous grounding and stochasticity values; hex quantization is only a serialization convention for the header. The cascade makes the precedence explicit: exhausted error budget dominates everything, hard grounding or stochasticity failures force regeneration, and the warn band is reached only after those guardrails are satisfied. In this benchmark, $\tau_{\text{warn}} = 0.12$, $\tau_{\text{crit}} = 0.22$, $g_{\min} = 0.33$, $\sigma_{\max} = 0.67$, and $E_{\max} = 2$, giving each question at most three drafts. Unlike opaque retry loops, each regeneration emits a fresh ASP header and the telemetry records which traces were rejected and why.

Failing answers are classified into regimes by their grounding profile.

- *Grounded partial*: citation matches and content overlaps the right material but does not cover enough required terms; controller target is *repair*.
- *Citation gap*: overlap ≥ 0.65 and term coverage ≥ 0.25 but wrong or missing citation; also *repair*.
- *Ungrounded*: insufficient grounding to justify refinement.

The first two are typically incomplete drafts that benefit from a second attempt; the third is typically reasoning-mode paraphrase that retrying will not fix.

4.4 Parameter Selection and Computational Overhead

ASP has multiple hand-tuned parameters: the linear weights in Eqs. 1–2, the ideal vector $\mathbf{v}^*$, the routing thresholds, and the error budget $E_{\max} = 2$. All values were set before benchmark execution from the qualitative rationale (grounding weighted toward source-language reuse, stochasticity toward fabrication risk); no parameter was adjusted in response to benchmark results. Sensitivity to these choices is the primary target for follow-on ablation work.

Computational overhead ASP header computation is $O(|\text{answer}| \cdot |\text{context}|)$ token comparison plus a fixed-size JSD evaluation. On the M-series Mac used in the benchmark, per-message sidecar overhead is sub-millisecond (a 30-token answer against 660 tokens of context completes in under 1 ms, dominated by tokenization). The wall-clock cost of a repair loop is dominated by re-generation time (on the order of seconds per retry for Qwen 0.8B), not by the sidecar itself. The protocol is therefore deployable in latency-sensitive settings; the dominant cost is whatever the underlying agent's generation cost is.

5 Experimental Setup

Reproducibility The ASP reference implementation, protocol documentation, and example telemetry are publicly available at `https://github.com/ananthasharma/argent-signaling-protocol`. The experimental benchmark harness, the 27-question Array/Ono evaluation set, the CUAD question-generation pipeline, the controller configuration (thresholds, weights, ideal vector $\mathbf{v}^*$), and the local GGUF model identifiers were developed in a separate research repository and are not part of the current public release. The Array BioPharma/Ono license agreement is a publicly filed SEC exhibit, and the CUAD corpus is publicly available [17].

5.1 Task and Document

The controller is tested using a publically available pharmaceutical license agreement between **Array BioPharma** and **Ono Pharmaceutical**. Legal agreements are a good stress test for grounded QA because they contain multi-branch definitions, conditional provisions, and cross-referenced terms that expose both retrieval and generation failures. The task is a 27-question document-grounded QA benchmark, with questions spanning six categories (Table 3). Each question specifies an expected answer, a set of 3–4 required terms, and citation references that identify the authoritative source text.

5.2 Retrieval, Generation, and Evaluation

Evidence is retrieved via TF-IDF over the agreement chunked at $\sim$220 words with paragraph overlap, with definition-aware bonuses (+1.0 for "*X* means" patterns, +0.5 for other definition syntax) to surface formal definitions. The top-$k = 3$ chunks are returned. This sparse approach is sufficient for single-section definitions but misses multi-branch clauses spanning non-adjacent paragraphs; the "Change in Control" definition is one such case. Domain-aware embedding retrofitting [18] is one mitigation. Each question is answered with a maximum token budget of 96 tokens; temperature is 0.1 on the first attempt and 0.05 on retries. Comparison is between a **Baseline** condition (one grounded answer attempt, no controller) and an **ASP controller** condition (the same initial generation followed by controller assessment, which decides whether to accept, warn, regenerate, or yield). An answer *passes* if it cites at least one chunk matching the expected citation *and* contains at least 60% of the required terms (case-insensitive). **This is a deliberately strict rubric**.

5.3 Models

The benchmark evaluates three local GGUF models chosen to span capability levels and produce qualitatively different failure modes: `Qwen3.5-0.8B-BF16` (compact instruction-tuned, follows instructions well, failures tend to be incomplete rather than fabricated); `dobby-8b-unhinged-q6_k` (8B parameters, not instruction-tuned for document QA, produces coherent grounded answers on a meaningful fraction of questions with incomplete-coverage failures); and `SmolLM3-3B-Q8_0` (moderate instruction following, alternates between grounded answers and uncited reasoning-mode paraphrase). Qwen and Dobby ran on MLX (Apple's optimized framework for Mac hardware); SmolLM3 used `llama.cpp`. The primary metrics are pass rate, citation match rate, mean required-term coverage, and controller action counts.

6 Results: Standalone Controller

This section reports standalone-controller results in three steps: a deep dive on the Array/Ono benchmark, two qualitative trajectory cards from that same benchmark, and a breadth test over 19 CUAD contracts.

6.1 Array/Ono Deep Dive

Table 4 reports per-model outcomes on the 27-question Array/Ono benchmark. Across the three models, the benchmark covers 81 model-question pairs. Aggregate passes move from 12/81 in baseline to 21/81 under ASP, with 10 fail-to-pass

Table 3: Question categories in the 27-question Array/Ono benchmark.

Category	Count
Definitions (Change in Control, Field, First Commercial Sale, etc.)	7
License scope (development, manufacturing, sublicense, grant-back)	6
Termination and wind-down	4
General provisions (governing law, amendment, assignment, etc.)	6
Governance (JDRC establishment and duties)	2
Risk allocation (recall control, indemnification)	2

recoveries and 1 regression. The controller applies 58 repair interventions, 12 containment interventions, and 11 clean pass-throughs. *Pass-through* is a controller decision (the routing cascade of Eq. 6 yields EXECUTE on the first draft: $g \geq g_{\min}$, $s \leq \sigma_{\max}$, $d \leq \tau_{\text{warn}}$); a *pass* is a benchmark outcome (citation match plus $\geq 60\%$ required-term coverage). These are distinct events by construction, but on this benchmark every pass-through also satisfied the benchmark criterion (all 11 pass-throughs are also passes) so the two counts coincide here.

ASP plays two distinct operational roles across the three models. On Qwen and Dobby, both producing 0% ungrounded outputs, ASP acts as a *repair controller*: Qwen passes rise from 3/27 to 9/27 and Dobby from 9/27 to 12/27. The Qwen cell and the aggregate (12/81 $\rightarrow$ 21/81, $p = 0.006$) are individually significant; the Dobby cell is not significant at $n = 27$ and should be read as part of the aggregate signal. Qwen's 29-point coverage gain (36.7% to 65.4%) is the strongest single effect; Dobby shows a small citation-precision tradeoff (70.4% to 66.7%) and one regression where regeneration broadens coverage by drawing on adjacent chunks.

On SmolLM3 (44.4% ungrounded), ASP improves nothing the end user sees as a passed answer (0/27 either way); its operational value is preventing the 12 ungrounded answers from being retried into more polished fabrication. The 15 repair attempts on grounded-but-incomplete drafts produce no additional passes; the 12 containment events are the result that matters for governance.

6.2 Signal Trajectory Analysis

Two representative controller trajectories from the Array/Ono benchmark make the signal values concrete. Required terms are listed; matched terms appear in **bold**. The first illustrates repair on Qwen; the second, containment on SmolLM3.

Q: When does Ono indemnify Array? Qwen 0.8B (repair $\rightarrow$ **pass**)

Required terms: Third Party Claim, Ono Territory, clinical studies, gross negligence or willful misconduct

Baseline Response `@G=F @S=0 d=0.176 cov=50%`

Ono shall indemnify Array against **Third Party Claims** arising from the use, marketing, distribution, or sale of Array Products, injury to patients in **Clinical Studies**, or breaches of representations and warranties by Array, as defined in Article 15 [Chunk 281].

After ASP Controller `@G=F @S=1 d=0.278 cov=75%`

Ono shall indemnify Array against **Third Party Claims** arising from the use, marketing, distribution, importation, or sale of Array Products, injury to patients in **Clinical Studies**, or breaches of representations and warranties by Array, as defined in Article 15 [Chunk 281]. Additionally, Ono shall indemnify Array against claims resulting from the **gross negligence or willful misconduct** of an Array Indemnitee [Chunk 280].

Outcome: PASS. Repair added Chunk 280 (exception clause), recovering one missing required term. "Ono Territory" was never recovered.

Table 4: Per-model outcomes on the 27-question Array/Ono benchmark. Rates are percentages. Action profile reports repair interventions (R), containment interventions (C), and clean pass-throughs (P). p is the one-sided McNemar exact test on paired baseline-vs-ASP outcomes; only Qwen and the aggregate clear $\alpha = 0.05$ at this sample size.

Model	Backend	Pass	Cite	Coverage	p	Actions
Qwen 0.8B	MLX	11.1 → 33.3	55.6 → 59.3	36.7 → 65.4	**0.016**	24R / 0C / 3P
Dobby 8B	MLX	33.3 → 44.4	70.4 → 66.7	48.8 → 54.3	0.19	19R / 0C / 8P
SmolLM3 3B	llama.cpp	0.0 → 0.0	18.5 → 18.5	24.7 → 25.6	—	15R / 12C / 0P

Q: What counts as a Change in Control? SmolLM3 3B (containment → **yield**)

Required terms: fifty percent, majority, merger, substantially all

Baseline Response `@G=3 @S=A d=0.620 cov=0%`

⟨think⟩ . . . A Change in Control would typically involve a shift in ownership or management. . . [reasoning-mode paraphrase, no chunk citations, no source terms]

After ASP Controller (attempt 3) `@G=2 @S=B d=0.650 cov=0%`

⟨think⟩ . . . Change in Control generally refers to an acquisition or transfer of control. . . [continued reasoning-mode paraphrase, no chunk citations]

Outcome: YIELD. Signals degraded across attempts: @G dropped 3→2, @S rose A→B, drift increased 0.620→0.650. The model stayed in uncited reasoning mode, producing plausible-sounding text with no connection to the retrieved chunks. Correctly routed to containment.

Qwen's high @G and low @S keep it in the repair regime; SmolLM3's degraded signals (@G in 2–3, @S in A–B) trigger containment.

6.3 Breadth Test: 19 CUAD Contracts, 143 Questions

We then test whether the same controller behavior generalizes beyond a single agreement. We extend the standalone benchmark to a stratified 19-contract sample drawn from the CUAD corpus [17], balanced across four file-size quartiles and 14 contract types (license, services, distribution, sponsorship, manufacturing, hosting, outsourcing, joint venture, and others). Questions are auto-generated from CUAD's per-contract clause annotations, yielding 143 questions across 10 categories: IP (30), termination (23), general provisions (17), risk allocation (16), restrictions (15), financial (13), metadata (9), definitions (7), governance (7), and rights (6). Each question is answered by the same three models under both conditions, totaling 429 model-question runs. The same controller configuration, retrieval pipeline, and pass criteria are reused unchanged; there is no per-document tuning.

Three findings dominate. First, the *utility* gain shrinks: from +11.1 absolute pass-rate points on the Array deep dive to +2.1 points aggregate on CUAD (5.8% → 7.9%). Per-cell movement is mostly not significant at $n = 143$ (Dobby $p = 0.25$, SmolLM3 $p = 0.50$; only Qwen reaches $p = 0.016$); the aggregate one-sided McNemar exact is $p = 0.002$, but per-model cells should not be read as individually demonstrated improvements. Second, the *safety* profile is preserved: zero regressions across 429 runs, and the single Array-Dobby regression is not reproduced. Third, the controller's operational regime inverts: repair-dominant on Array (58R / 12C) and containment-dominant on CUAD (9R / 395C, 92.1% containment). On heterogeneous contracts the controller's value shifts from *repair* to *containment* — the QA-improvement effect mostly evaporates while the safety and failure-mode-legibility properties transfer. All 9 issued repair attempts converted fail-to-pass.

Backend note for Dobby Dobby was evaluated on `llama.cpp` for the CUAD benchmark; the MLX path produced deterministic token-soup output across all 143 CUAD questions despite working on the Array document with the same model file (29 unique 80-char answer prefixes across 143 outputs). This is treated as a property of the inference stack rather than the protocol — ASP's drift and grounding signals would have flagged such failures regardless.

7 Results: Multi-Agent Pipeline

The focus here is to test if ASP prevents semantic drift *across agent boundaries*, rather than only within a single agent's retry loop. We evaluate a two-agent pipeline where an upstream retrieval agent (Agent A) answers a question and a downstream decision agent (Agent B) must make a risk assessment based on that answer.

Table 5: Standalone controller on the CUAD breadth benchmark (19 contracts, 143 questions $\times$ 3 models = 429 runs). Rates are percentages. Action profile: repair (R), containment (C), pass-through (P). p is the one-sided McNemar exact test; only Qwen and the aggregate reach $\alpha = 0.05$.

Model	Backend	Pass	Cite	Coverage	p	Actions	FtP / Reg
Qwen 0.8B	MLX	7.0 → 11.2	30.1 → 32.2	25.9 → 35.3	**0.016**	6R / 127C / 10P	6 / 0
Dobby 8B	llama.cpp	7.0 → 8.4	25.9 → 25.9	31.5 → 35.0	0.25	2R / 131C / 10P	2 / 0
SmolLM3 3B	llama.cpp	3.5 → 4.2	9.8 → 9.1	22.6 → 28.3	0.50	1R / 137C / 5P	1 / 0
Aggregate		**5.8 → 7.9**	21.9 → 22.4	**26.7 → 32.9**	**0.002**	**9R / 395C / 25P**	**9 / 0**

7.1 Pipeline Design

Agent A is the same retrieval QA agent from the controller benchmark. Agent B receives Agent A's output and must classify the provision as HIGH/MODERATE/LOW RISK with a one-sentence justification grounded in the cited terms. We compare two conditions quantitatively. **Baseline**: Agent B receives Agent A's raw text with no quality metadata. **ASP gated**: the sidecar intercepts Agent A's output; if grounding falls below $g_{\min}$, stochasticity exceeds $\sigma_{\max}$, or drift exceeds τ_{crit}, the output is *blocked* and Agent B receives an escalation message instead.[2]

7.2 Pipeline Results

Table 6 and Figure 3 summarize the pipeline benchmark on SmolLM3 (the model with the mixed failure profile, 44.4% ungrounded). In the baseline condition, all 27 outputs (including the 14 ungrounded ones) pass directly to Agent B, which then makes risk assessments based on fabricated or poorly grounded upstream content. In the ASP-gated condition, the sidecar blocks 24 of 27 outputs (including all 14 ungrounded ones); the 3 outputs that pass through are the least-drifted grounded answers.

7.3 Why SmolLM3 Only, and the Scope of This Claim

The pipeline benchmark is restricted to SmolLM3 because Qwen and Dobby produce 0% ungrounded outputs standalone; applying the sidecar to them would trivially pass 100% through. SmolLM3 is the only condition where the gate is exercised — which is also the reason for caution: *this section is a single-model existence proof.* The headline number, 100% of 14 ungrounded outputs blocked, is conditional on those 14 samples from one model on one document. Generalising would require an adversarial protocol that manufactures ungrounded outputs on Qwen and Dobby, or a multi-model pipeline with at least one weak link; both are follow-on work. The 88.9% overall block rate reflects SmolLM3's failure profile, not the sidecar policy.

7.4 Governance Implication

On this case study, without ASP, 51.8% of upstream outputs propagated ungrounded content to a downstream decision agent; with the ASP sidecar, zero ungrounded outputs reach the downstream agent. The sidecar does not make SmolLM3 more capable (Agent A's pass rate remains 0%), but it prevents unreliable outputs from contaminating downstream decisions. This illustrates the multi-agent version of the same principle demonstrated in the standalone benchmark: ASP changes how failure propagates, not how often models succeed. Whether the same effect holds across other weak-agent configurations is an empirical question this paper does not answer.

8 Operational Interpretation

8.1 Failure Legibility

The benchmark supports two operational claims:

- ASP produces meaningful aggregate gains (12/81 to 21/81 passes, 10 recoveries).
- ASP changes controller behavior in a way that matches the model's failure regime.

[2] In a qualitative probe in which Agent B received the output together with the full ASP header (no hard gate). Agent B's judgments shifted toward hedged language when the header showed low grounding, suggesting that header visibility alone can change downstream behaviour.

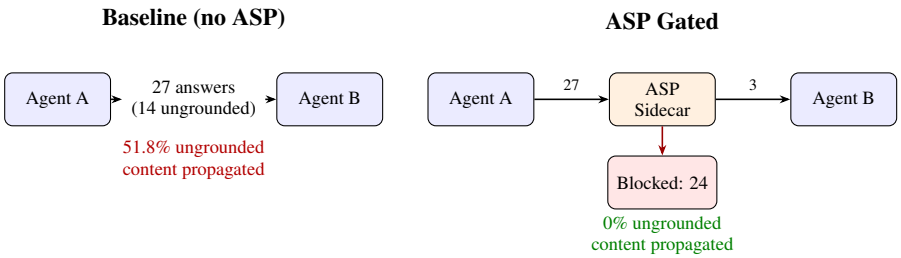

Figure 3: Pipeline comparison. **Left**: without ASP, all 27 outputs (including 14 ungrounded) reach Agent B. **Right**: the ASP sidecar blocks 24 outputs, allowing only 3 grounded answers through. Zero ungrounded content reaches Agent B.

Table 6: Pipeline benchmark results (SmolLM3, 27 questions, risk assessment task).

Metric	Baseline	ASP gated
Agent A outputs reaching Agent B	27/27 (100%)	3/27 (11.1%)
Ungrounded outputs reaching Agent B	14/27 (51.8%)	0/27 (0%)
Blocked by sidecar	0	24/27 (88.9%)
Escalated to human review	0	24

A human supervisor reviewing a batch of 81 outputs without ASP sees only raw results and has no structured way to prioritize which failures to investigate first. With ASP, the same batch decomposes into 11 clean passes, 58 repair attempts (10 of which recovered to passing), and 12 containment events (none worth additional effort). This makes the system's own failure state legible: Qwen and Dobby outputs are worth repairing; SmolLM3 produces no additional passing answers but its 12 ungrounded outputs are halted before they consume further compute. The controller separates those cases rather than treating all non-passing drafts as equivalent.

8.2 Coverage Gains and Limits

The signal formulas in Eqs. 1–2 and the fallback rule in Eq. 3 are intentionally simple, not learned classifiers — the signals must be auditable without a calibration dataset. The controller therefore cannot distinguish subtle failure modes (e.g., plausible-but-wrong from clearly nonsensical) unless they also diverge in token overlap and citation profile. For this benchmark the separation holds: Qwen coverage rises 36.7% to 65.4% (29 points); Dobby 48.8% to 54.3% with 4 new passes. A reviewer receiving an answer with 65% term coverage is better positioned than one receiving 37%. Whether the separation holds for more nuanced tasks remains open.

9 Limitations

The scope of evidence presented here is narrow in several respects:

- The pass rubric is strict but synthetic. It depends on citation match and required-term coverage rather than human judgment of answer usefulness.
- **Signal/criterion correlation** The controller rewards context overlap (o_c, Eq. 1); the pass criterion rewards required-term coverage ($\geq 60\%$). These are not the same set, but they are correlated by construction (retrieved chunks contain the language the required terms are drawn from) so a regeneration that maximises source-language reuse tends to recover more required terms. Qwen's 29-point coverage gain is therefore partly the controller selecting drafts favourable under a proxy of the pass criterion, not solely independent quality improvement. Citation match is an orthogonal pass requirement, which bounds the effect, but a fully independent rubric (e.g., human judgement) would be needed to rule it out.
- The signal computation (Eqs. 1–2) uses fixed hand-tuned weights and a hand-picked ideal vector $\mathbf{v}^*$. The thresholds were not perturbed in an ablation; their sensitivity is not characterised in this work.
- The full protocol described in the extended (arXiv) version of this paper additionally includes a stochasticity-modulated centroid update, an adaptive assumption-decay rate, and an active→decayed→quarantined assumption lifecycle for multi-turn deployments. None of these multi-turn mechanisms are evaluated here: the camera-ready benchmark runs bounded at-most-3-draft trajectories and exercises only the per-draft scoring path; their effect on extended dialogues remains future work.

Acknowledgments

The author thanks the anonymous XTAI 2026 reviewers for substantive feedback that sharpened the failure-legibility framing and the closed-API estimator discussion. Any remaining errors are the author's.

References

[1] Qingyun Wu et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Proceedings of the First Conference on Language Modeling*, 2024.

[2] Guohao Li et al. CAMEL: Communicative agents for mind exploration of large scale language model society, 2023. arXiv:2303.17760.

[3] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017.

[4] Saurav Kadavath et al. Language models (mostly) know what they know, 2022. arXiv:2207.05221.

[5] Alejandro Barredo Arrieta et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[6] Elham Tabassi. Artificial intelligence risk management framework (AI RMF 1.0). Technical Report NIST AI 100-1, National Institute of Standards and Technology, 2023.

[7] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daume III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[8] Matthew Arnold et al. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5):6:1–6:13, 2019.

[9] Anantha Sharma, Swetha Devabhaktuni, and Eklove Mohan. The Argent framework: A paradigm to compose, orchestrate, and govern enterprise AI agents, 2025. SSRN preprint.

[10] Aman Madaan et al. Self-refine: Iterative refinement with self-feedback, 2023. arXiv:2303.17651.

[11] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. arXiv:2303.11366.

[12] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. arXiv:2302.09664.

[13] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Zhang, Jinfeng Yi, et al. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[15] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37 (1):145–151, 1991.

[16] Dominik M. Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2003.

[17] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An expert-annotated NLP dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. arXiv:2103.06268.

[18] Anantha Sharma. Embedding retrofitting: Data engineering for better RAG. SSRN preprint, 2026. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6064714`.