drones

MDPI

Article

Evidence-Carrying Mission Admission Contracts for Natural-Language UAV Task Submission

Zhiwei Huang [1,2], Gang Wei [2], Gang Wang [2,*], Xuan Liu [1,2], Haolun Sun [1,2], Xiaoyang Han [2] and Hui Yuan [2]

[1] Graduate School, Air Force Engineering University, No. 1 Changle East Road, Baqiao District, Xi'an 710051, China; hzw_1127@outlook.com (Z.H.); 19552237851@163.com (X.L.); 18644884707@163.com (H.S.)

[2] College of Air and Missile Defense, Air Force Engineering University, No. 1 Changle East Road, Baqiao District, Xi'an 710051, China; weigang_afeu@163.com (G.W.); dodogg189888@163.com (X.H.); huiyuan728@163.com (H.Y.)

* Correspondence: 18972020859@163.com

Highlights

What are the main findings?

- EAMSR turns natural-language UAV task submission into an auditable mission-admission decision.
- Unwarranted ADMIT decisions are eliminated on EAMSR-Bench; remaining errors stay outside direct admission.

What are the implications of the main finding?

- LLM-generated UAV missions should be governed as candidates, not trusted as ready-to-plan specifications.
- Pre-execution admission creates a verifiable handoff before downstream UAV planning and runtime safety layers.

Abstract

Natural-language interfaces now influence mission generation in UAV planning pipelines, where a feasible plan may still rest on unsupported completions, unauthorized relaxations, or consequence-changing repairs. We present EAMSR, an evidence-carrying Mission Admission Contract framework that treats a submitted mission as an admission contract, not a text-to-specification output. A large language model proposes candidate clauses and bounded refinements but does not decide admission; the governance layer links hard clauses to evidence and authorized sources, isolates unsupported semantic increments, checks mission-consequence compatibility, and accepts only candidates with a backend task-level witness. The final decision (ADMIT, CLARIFY, or REJECT) carries an audit trail linking language anchors to governance and backend outcomes; backend search failures that do not establish model-level infeasibility return CLARIFY, not REJECT. On EAMSR-Bench (120 tasks, six scenarios, six risk types), EAMSR achieves 100.0% binary admission accuracy (Acc_{bin}) and 93.3% three-class accuracy (Acc_{adm}), with no unwarranted ADMIT decisions among 78 non-admissible cases (0/78; 95% Clopper–Pearson upper bound, 3.8%); residual errors lie on the CLARIFY–REJECT boundary. On an independent external test set (48 tasks), EAMSR achieves 89.6% three-class and 100.0% binary accuracy with no unwarranted admissions. A deterministic non-LLM variant using the same governance and backend layers also attains zero unwarranted admissions but lower accuracy, indicating that LLM generation improves semantic coverage while governance and backend checks determine admission. The evaluation is limited to pre-execution mission admission under stated assumptions and does not constitute flight-safety assurance.

Check for updates

Academic Editor: Peihu Duan

Received: 22 June 2026
Revised: 17 August 2026
Accepted: 23 August 2026
Published: 16 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: unmanned aerial vehicles; natural language interface; human–UAV interaction; task admission; mission admission contract; large language models; mission planning; task-level validation

1. Introduction

Natural-language interfaces have moved beyond auxiliary interaction in UAV systems; they now participate in the mission-generation chain. Once a language input is fixed as a task condition, an interpretation error propagates into the planning process. An unsupported completion becomes a pre-execution mission assumption, not just a textual deviation. Natural-language UAV task submission requires an admission decision, beyond semantic parsing alone.

Existing studies have made natural-language task requests structurally expressible. Transformations from language to task specifications, temporal logic, or planning models give raw requests checkable forms [1–11], but a structured output is not automatically admissible. Planner diagnosis and specification repair can show that a modified task has a feasible plan [12–18], yet the existence of a plan does not supply missing evidence, authorize constraint relaxation, or prevent consequence-changing repairs from concealing semantic shifts. Safety gates can block dangerous actions or unacceptable plans before execution [19–26], but they operate on transformed artifacts rather than on the clause-formation process, so a plan that passes safety checks may still contain unsupported hard clauses, unauthorized relaxations, or unresolved consequence changes. Translation correctness, planning feasibility, and operational safety each address part of the chain, but none implies admission integrity.

This requirement calls for a decision object distinct from text, specifications, and plans: one that places clause provenance, evidence status, modification authority, mission consequences, and backend executability within a single admission context [27–30]. We define this object as a Mission Admission Contract (MAC): an auditable admission commitment before a task enters the planning chain, not a replacement for translators, planners, or safety gates. We propose EAMSR, an Evidence-Carrying Mission Admission Contract framework built on the MAC. EAMSR restricts the LLM to generating candidate clauses and bounded refinement suggestions, while placing admission responsibility in the governance layer and the backend task model. An ADMIT decision requires evidence, authority, consequence, and backend-witness checks, not merely a well-formed LLM specification; requests whose admission conditions cannot be closed are returned as CLARIFY or REJECT with an audit trail.

The contributions of this paper are as follows.

First, we formulate natural-language UAV task submission as mission admission and introduce the MAC as an evidence-carrying admission object that makes clause provenance, evidence status, and modification authority preconditions for entering the planning chain.

Second, we develop a conjunctive admission architecture that separates untrusted candidate generation from evidence, authority, consequence, and task-level witness verification, so that unverified semantic increments and unauthorized relaxations cannot silently become hard mission obligations.

Third, we establish a controlled evaluation protocol covering binary admission correctness, three-class remediation decisions, mechanism diagnostics, external-authoring robustness, and backend consistency.

The scope of this paper is bounded between task submission and task planning. The claims hold under stated assumptions: given a valid trusted context, governance

policy, evidence base, and backend task model, EAMSR enforces the specified admission conditions before tasks enter the planning chain.

2. Related Work

Prior work on natural-language UAV interaction has addressed three stages of the mission-generation chain: language-to-specification translation, plan repair, and pre-execution safety gating. Existing studies can derive mission regions, waypoints, and constrained paths from compound commands [1], and can connect LLM planning to mission orchestration pipelines that include authentication and human approval [7,11,31]. Authentication confirms the requesting subject, and human approval can confirm a final proposal, but neither explains how task clauses in the proposal are formed from the original instruction. The key question is whether a model-transformed task request has admission integrity before entering the planning chain.

Formal translation and symbolic planning improve the checkability of task requests but do not by themselves prove admission integrity [5,8,32–35]. Their decision object is specification correctness: nl2spec provides span-level traceability for translation revision [2]; ClarifySTL identifies underspecified requirements without deciding whether alternative completions change mission consequences [3]; SPAR constructs PDDL planning models for operator requests, not for admission decisions [4]. Planner-based repair methods such as DUPLEX and UAV-NL-STL-Repair restore executability through diagnosis or MILP feedback [12,13], but traceability, formal completeness, and replannability do not authorize clause modification: unsupported semantic increments and repairs that cross consequence-compatibility sets can still produce executable but inadmissible missions.

Operational safety methods adopt a different decision object: whether a final command or plan is dangerous, feasible, or rule-compliant. SafeGate adjudicates safety before code generation [19]; RoboGuard enforces temporal-logic guardrails on LLM plans [20]; UAV-specific studies verify spatial, environmental, and operational constraints through probabilistic models and rule-based verification [21,22]; complementary guardrails and verification frameworks for LLM-driven robots address related checks [16,19,20,23–26,36]. Because these methods operate on transformed artifacts, a plan that passes safety checks may still contain unsupported completions or unauthorized relaxations.

Assume–guarantee and contract-based approaches refine and repair unrealizable robotic mission specifications [37–41], but their starting point is an already formalized and trusted specification, and none targets natural-language grounding or admission-level evidence tracing. What remains missing is a joint admission decision over evidence and authority, untrusted-increment isolation, consequence compatibility, and backend executability.

Tables 1 and 2 compare EAMSR with the most closely related methods across ten capability dimensions and delineate conceptual boundaries between EAMSR components and prior concepts, focusing on specific systems.

Principal methodological novelty. Among the methods directly compared in Table 1, none exposes all of these elements as first-class admission variables. The principal novelty of EAMSR is the integration of these mechanisms into a single conjunctive admission predicate for natural-language UAV task submission. The MAC representation adapts provenance-tracked specification concepts; the proof-obligation gate extends policy-based authorization with evidence, authority, mutability, and untrusted-isolation checks; the authority-bounded refinement relation builds on assume–guarantee contract refinement; the mission-consequence compatibility mechanism is a task-level analog of semantic-preserving specification refinement; and the backend plan witness is a task-level analog of plan validation. Each individual component has precedent in prior work, but their conjunctive combination for natural-language UAV mission admission is new.

Table 1. Comparison of EAMSR with closely related methods across ten capability dimensions. "Not addressed" indicates that the dimension is not explicitly modeled or tracked by the method.

Dimension	nl2spec	SafeGate	DUPLEX	CROME	EAMSR
NL grounding	Text-to-spec translation	Safety check on generated code	Repair of infeasible specs	Refinement of formal specs	Anchor extraction from NL
Clause provenance	Span-level trace	Not addressed	Not addressed	Assumed from formal input	Source-class tagged per clause
Evidence status	Not addressed	Not addressed	Not addressed	Not addressed	Three-valued (Y/N/U) with authority and freshness
Authority	Not addressed	Not addressed	Not addressed	Not addressed	Five-level hierarchy with delegation
Mutability	Not addressed	Not addressed	Not addressed	Contract-based refinement	Five-mode taxonomy (immutable to revocable)
Ambiguity handling	Translation revision	Plan rejection	Diagnosis-based repair	Contract refinement	CLARIFY with remediability test
Semantic preservation	Spec correctness	Safety constraint satisfaction	Executability restoration	Assume–guarantee refinement	Non-transitive consequence compatibility
Backend executability	Not addressed	Not addressed	Planner-based repair	Not addressed	Task-level plan witness
Audit output	Translation trace	Safety verdict	Repair log	Refinement history	Full admission audit trail
UAV-specific validation	Domain-agnostic	Robotic systems	UAV missions	Robotic missions	UAV ontology and backend model

Table 2. Conceptual boundaries between EAMSR components and related prior concepts.

EAMSR Component	Related Prior Concept	Essential Distinction
Mission Admission Contract (MAC)	Assurance case / proof-carrying artifact	Assurance cases provide structured argument–evidence reasoning that a system is safe, whereas a MAC records whether a specific mission request satisfies admission conditions before planning.
Proof-obligation gate	Policy-based authorization	Policy engines check access permissions, whereas the gate also requires clause-level evidence, mutability compliance, and untrusted-semantic-isolation checks.
Backend witness	Standard plan validation	Plan validation verifies that a plan satisfies a specification, whereas the witness closes executability only after clause governance and consequence compatibility have been established.
Mission-consequence compatibility	Semantic-preserving specification refinement	Semantic preservation requires preserving a specified logical or behavioral relation, whereas compatibility is a non-transitive tolerance relation that permits bounded changes in non-critical dimensions while vetoing changes in critical ones.
Evidence-carrying clauses	Provenance-tracked requirements	Provenance records where a clause came from, whereas evidence-carrying clauses also require that each hard clause be supported by admissible evidence with verified authority and freshness.

Table 2 delineates five conceptual boundaries that clarify what each EAMSR component is and what it is not, relative to the closest prior concept.

3. Materials and Methods

Natural-language UAV task submission faces three admission risks: operator ambiguity or infeasible requirements, LLM-introduced unsupported assumptions, and missions that appear reasonable at the language level while violating backend constraints. To address these risks, we propose EAMSR, an evidence-carrying Mission Admission Contract framework for natural-language UAV task submission. EAMSR instantiates a natural-language mission instruction as a Mission Admission Contract (MAC), and performs evidence tracing, authority checking, untrusted semantic increment isolation, mission-consequence screening, and backend plan-witness checking before the task enters the planning chain [27–30,38]. Figure 1 shows the overall framework architecture and decision flow.

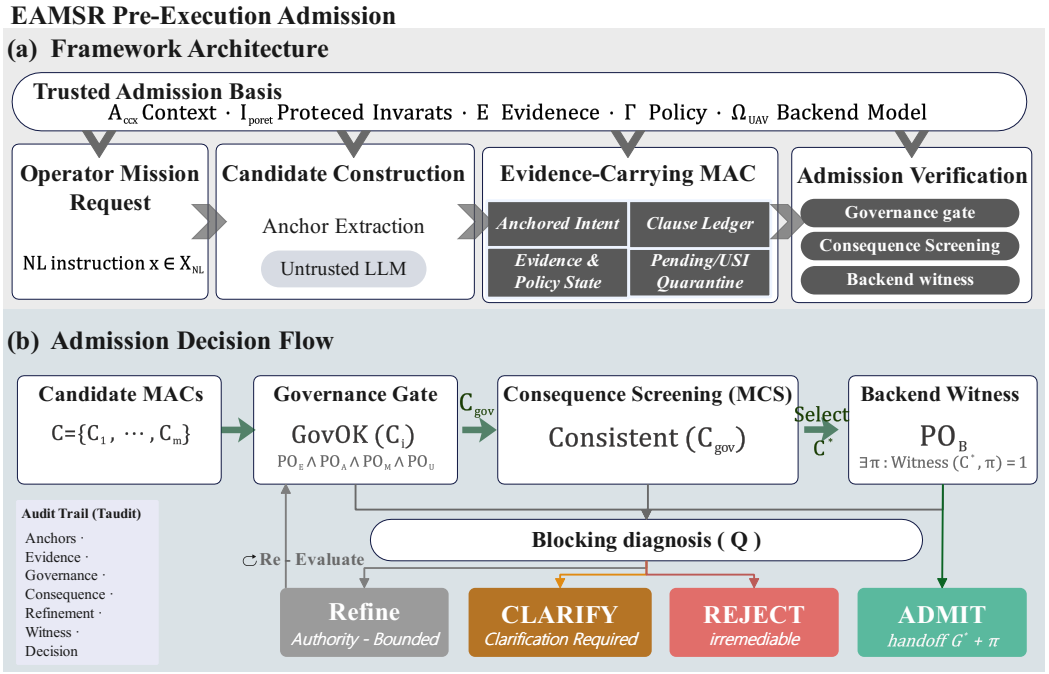


Figure 1. Overview of the EAMSR framework. Natural-language UAV mission requests are instantiated as evidence-carrying Mission Admission Contracts (MACs). A governance proof-obligation gate enforces evidence, authority, mutability, and untrusted-semantic-isolation checks; governance-admissible candidates are screened through mission-consequence compatibility; a backend plan witness closes the selected candidate. Final admission requires all three stages.

The system output is

$$decision \in \{\text{ADMIT}, \text{CLARIFY}, \text{REJECT}\}. \quad (1)$$

ADMIT means the task request has formed an admission contract satisfying evidence, authority, consequence, and backend-witness requirements; CLARIFY means that ambiguity may change operational consequences, that an operator-authorized modification is required, or that backend closure remains inconclusive; REJECT means the request conflicts with an immutable or protected condition, or that task-level infeasibility is conclusively established, within the currently declared trusted context, governance policy, and backend model (Section 3.5).

3.1. Mission Admission Contract and Trust Boundary

A natural-language mission instruction in a UAV scenario is a joint request over system capability, safety boundaries, and mission consequences. We first define the Mission Admission Contract to clarify the boundary among the operator request, trusted environment, protected constraints, and system admission responsibility.

Let the operator's natural-language mission instruction be $x \in \mathcal{X}_{\mathrm{NL}}$. The instruction may contain mission objectives, target areas, observation windows, execution order, payload requirements, communication-return requirements, soft preferences, and abort conditions. EAMSR constructs a base Mission Admission Contract C_0 from x:

$$C_0 = \langle A_{\mathrm{ctx}}, I_{\mathrm{prot}}, G_{\mathrm{req}}, \Gamma, \mathcal{E}, \Omega_{\mathrm{UAV}} \rangle. \tag{2}$$

In this definition, C_0 is the base contract derived from the original instruction; A_{ctx} denotes the trusted UAV context, including the UAV capability model, initial state, mission environment, map, target locations, weather, communication coverage, time windows, and payload status. I_{prot} denotes protected operational invariants, including return reserves, no-fly zones, geofences, altitude limits, payload limits, operating rules, and emergency-abort conditions. G_{req} denotes the mission-request obligations proposed by the natural-language instruction. Γ denotes the governance policy, including source verification, authority levels, and mutability rules. $\mathcal{E}$ denotes the evidence base, including text anchors, trusted context facts, protected constraints, and system default rules. Ω_{UAV} denotes the UAV task-level backend model. Candidate contracts C_i (Equation (15)) are generated from C_0 by the LLM and form the candidate set $\mathcal{C} = \{C_1, \ldots, C_m\}$ where $m \leq B_c$, as duplicate candidates are discarded (Section 3.2).

When admission succeeds, the system selects a contract C^* and extracts its admitted mission obligations $G^* = G(C^*) = \langle H^*, S^*, P^* \rangle$, where H^* denotes the hard goals and hard constraints, S^* the soft preferences, and P^* the pending clauses. A task-level plan witness π must satisfy the hard obligations:

$$\pi \vDash A_{\mathrm{ctx}} \wedge I_{\mathrm{prot}} \wedge H^* \wedge \Omega_{\mathrm{UAV}}. \tag{3}$$

Soft preferences influence candidate selection (Equation (32)) but are not backend proof obligations; pending clauses cannot enter H^* and therefore do not affect witness closure. P^* is retained in the selected contract and audit trail for completeness but does not constitute a downstream obligation; any pending clause with consequence relevance must trigger CLARIFY before admission can proceed. The EAMSR admission objective is defined as

$$x \longmapsto C \longmapsto (G^*, \pi, \mathit{decision}), \tag{4}$$

subject to

$$\mathit{decision} = \mathrm{ADMIT} \Rightarrow \mathrm{AdmissionReady}(C^*, \mathcal{C}_{\mathrm{gov}}) = 1 \wedge \mathit{Witness}(C^*, \pi) = 1. \tag{5}$$

The trust boundary of EAMSR is limited to the pre-execution admission layer between task submission and task planning. This layer handles task goals, spatial regions, time windows, payloads, communication, energy, return reserves, restricted airspace, and abort conditions. Low-level path planning, continuous obstacle avoidance, flight control, and runtime safety monitoring are delegated to downstream execution modules.

3.2. Contract Instantiation for Natural-Language Mission Requests

The first admission risk of natural-language UAV tasks arises from semantic grounding. A single instruction may contain goals, regions, time constraints, payload requirements, and communication requirements. These contents fall into three categories: evidence-supported, clarification-required, and LLM-introduced completions. Contract instantiation converts the natural-language request into clause-level, evidence-carrying, and auditable candidate mission obligations.

EAMSR extracts requirement anchors from the instruction x:

$$\mathcal{A} = \{a_1, a_2, \ldots, a_n\}. \tag{6}$$

Each anchor a_i is a text span with independent mission meaning. The extraction procedure operates in four steps: (i) span detection segments the instruction at linguistic boundaries (clauses, conjunctions, conditionals) to identify candidate spans; (ii) ontology slot assignment maps each span to a clause type in the UAV ontology (temporal, spatial, communication, payload, etc.); (iii) modality identification labels each span as hard, soft, optional, or conditional based on linguistic cues (imperative, "if possible", "as needed"); (iv) validation checks whether each span has a clear referent in A_{ctx} and a deterministic ontology relation. All four steps use deterministic linguistic rules and ontology lookups; no LLM is involved in anchor extraction. Anchor extraction is implemented deterministically and is treated as trusted preprocessing in the present evaluation; extraction correctness is therefore an explicit assumption. Spans that fail validation (e.g., vague referents, unresolved negation scope, or cross-clause dependencies without explicit conjunction) are marked as *pending* and excluded from clause generation. This procedure prevents the LLM from interpreting its own completions as user-confirmed requirements.

Under the UAV mission ontology $\mathcal{O}_{\text{UAV}}$ and trusted context A_{ctx}, the LLM generates one candidate contract clause for each validated anchor:

$$\kappa_i = \text{LLM}_\theta(a_i, \mathcal{O}_{\text{UAV}}, A_{\text{ctx}}). \tag{7}$$

The set of all candidate clauses for a single generation pass is denoted $\mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_n\}$, where n is the number of validated anchors.

To produce multiple candidate contracts, EAMSR performs up to B_c repeated LLM invocations under the same decoding settings. In the evaluated configuration, temperature is 0; candidate variation therefore arises only from residual model-serving nondeterminism or alternative structured assignments returned for ambiguous anchors. Each invocation generates a complete clause set $\mathcal{K}^{(b)}$ from the same anchor set $\mathcal{A}$; the clause sets are then assembled into candidate mission obligations G_i and candidate MACs C_i (Equations (14) and (15)). The resulting candidate set is $\mathcal{C} = \{C_1, \ldots, C_m\}$, where $m \leq B_c$. Each generation attempt consumes one unit of the candidate budget B_c; duplicate candidates (identical clause sets) are detected by hash comparison, discarded, and recorded in the audit trail. Consequently, the final candidate set satisfies $0 \leq |\mathcal{C}| \leq B_c$ (the lower bound is reached only if all generation attempts produce unparseable responses, which triggers *decision* $\leftarrow$ CLARIFY), and the total number of LLM invocations remains bounded by $B_c + B_r$ (Property 4). In the evaluated configuration, $B_c = 5$ was selected based on a pilot study on a separate 20-task development subset: increasing B_c beyond 5 yielded diminishing returns in coverage (fewer than 2% additional governance-admissible candidates), while $B_c < 3$ occasionally left no governance-admissible candidate for tasks where a single LLM pass produced an incompatible clause assignment. A candidate clause is defined as

$$\kappa = \langle id, anchor, type, \phi, mode, source, evidence, authority, mutability, trust \rangle. \tag{8}$$

The fields are as follows: *id* is the clause identifier; *anchor* is a text anchor or trusted-context pointer; *type* is the UAV clause family; ϕ is the formal task semantics; *mode* is hard, soft, optional, default, or pending; *source* is the source category; *evidence* is the evidence pointer; *authority* is the authority level; *mutability* is the allowed modification mode; and *trust* is the trust status.

The *source* field classifies the origin of each clause into five categories (Table 3), which determines its default authority level and mutability. User-explicit clauses carry the highest authority among user-originated sources, but remain subordinate to applicable regulatory constraints and are treated as immutable hard intent unless the user authorizes modification. When the user subsequently authorizes a modification to a previously immutable user-explicit clause, this authorization creates a new authoritative anchor (an updated request version) rather than retroactively mutating the original clause; the original clause is preserved in the audit trail, and the refined contract carries the new anchor with the user's explicit authorization. Clauses derived from trusted context (e.g., weather data, airspace maps, vehicle specifications) inherit the authority of their source. Clauses introduced by the LLM during candidate generation or repair are labeled as hypothetical and must be verified before entering hard obligations.

Table 3. Clause source categories, default authority, and mutability.

Source	Description	Default Mutability
user_explicit	Directly stated in the user instruction	immutable (hard)
user_implied	Inference from user instruction via deterministic ontology rule	protected (hard)
trusted_context	Derived from trusted context data	governance-defined
system_default	Applied when user input is silent on a parameter	overridable
llm_generated	Introduced by the LLM during candidate generation or repair	revocable

A clause κ may be labeled *user_implied* only if it can be derived from an explicit anchor a_i and a deterministic ontology rule $r \in \mathcal{R}_{\text{ontology}}$:

$$source(\kappa) = user_implied \Rightarrow \exists\, a_i,\ r \in \mathcal{R}_{\text{ontology}} :\ a_i \wedge r \models \kappa. \tag{9}$$

If no such derivation exists, the clause is labeled *llm_generated* with trust *hypothetical*, preventing unsupported LLM completions from hardening into protected obligations.

The evaluated UAV ontology $\mathcal{O}_{\text{UAV}}$ contains 8 clause families (Table 4), 42 normalized predicate templates, 15 relation types (including temporal precedence, spatial containment, resource dependency, and authority delegation), and 6 entity types (vehicle, waypoint, region, sensor, payload, communication node). The deterministic ontology rules $\mathcal{R}_{\text{ontology}}$ used for *user_implied* derivation comprise 28 rules, each expressed as a Horn-clause implication over the predicate vocabulary. The full ontology specification and rule set are available in the reproducibility package.

Table 4. UAV contract clause families.

Clause Family	Content Examples
Objective	inspection, search, imaging, delivery, return, return-to-home
Spatial	waypoints, regions, no-fly zones, geofences, altitude ranges
Temporal	deadlines, observation windows, mission order
Energy	battery, endurance, return reserve, resource consumption
Payload	camera, thermal imaging, LiDAR, payload capacity, sensor status
Communication	link coverage, real-time return, delayed upload, relay conditions
Contingency	abort, hover, return-to-home, emergency handling
Preference	priority, quality preference, optional targets

The governance layer computes ternary semantic support for each candidate clause:

$$Sup(\kappa) \in \{\mathrm{Y}, \mathrm{N}, \mathrm{U}\}. \tag{10}$$

In this classification, Y indicates evidence support, N indicates evidence exclusion or invalid source, and U indicates insufficient evidence. The three-valued support classification is defined relative to the single resolved highest-priority applicable evidence source after authority and freshness conflict resolution, ensuring mutual exclusivity. $Sup(\kappa) = \mathrm{Y}$ means that the resolved highest-priority applicable fresh evidence source supports κ. $Sup(\kappa) = \mathrm{N}$ means that the resolved highest-priority applicable evidence source contradicts κ, or no admissible evidence source remains after authority/freshness validation (invalid source). $Sup(\kappa) = \mathrm{U}$ means that no resolved admissible source supports or contradicts κ (insufficient evidence), so the clause cannot enter a hard obligation but may remain as a pending item. Evidence freshness is verified by comparing evidence timestamps against context-defined validity windows: weather data, communication coverage, and airspace restrictions are checked against their respective update intervals, while battery state and payload availability are treated as real-time context fields. When two trusted-context facts conflict (e.g., two weather sources report different wind speeds), the governance policy specifies a priority order: regulatory sources override commercial data services, and more recent readings override older ones at the same authority level. A clause can enter a hard mission obligation only if

$$Sup(\kappa) = \mathrm{Y}. \tag{11}$$

New statements generated by the LLM during candidate generation or repair are labeled as untrusted semantic increments. For an LLM suggestion p, its claim set is decomposed as

$$Claim(p) = Claim_{\mathrm{R}}(p) \cup Claim_{\mathrm{D}}(p) \cup Claim_{\mathrm{H}}(p). \tag{12}$$

In this decomposition, $Claim_{\mathrm{R}}(p)$ contains reused claims traceable to user anchors or trusted context; $Claim_{\mathrm{D}}(p)$ contains derived claims entailed by rules, backend models, or protected constraints; and $Claim_{\mathrm{H}}(p)$ contains hypothetical claims introduced only by the LLM without external evidence. Hard clauses in admitted mission obligations must satisfy

$$Claims(Hard(G^*)) \cap Claim_{\mathrm{H}} = \varnothing, \tag{13}$$

where $Claims(\cdot)$ extracts the set of atomic claims from a clause set, as defined in Equation (12). This constraint confines unverified assumptions to pending items, clarification requests, or the audit trail.

The *trust* field in the clause structure records the verification status of each clause, taking values from {trusted, derived, hypothetical, pending} (Table 5).

Table 5. Trust labels and admission semantics.

Trust Label	Meaning	Hard Obligation
trusted	Traceable to user input or verified trusted context	Allowed
derived	Entailed by rules, backend model, or protected constraints	Allowed
hypothetical	LLM-introduced without external evidence	Prohibited
pending	Awaiting evidence or clarification	Prohibited

After clause labeling, EAMSR assembles candidate clauses into candidate mission obligations:

$$G_i = \langle H_i, S_i, P_i \rangle, \tag{14}$$

where H_i is the set of hard goals and hard constraints, S_i is the set of soft preferences, and P_i is the set of pending clauses. The corresponding candidate MAC is

$$C_i = \langle A_{\text{ctx}}, I_{\text{prot}}, G_i, \Gamma, \mathcal{E}, \Omega_{\text{UAV}} \rangle. \tag{15}$$

EAMSR also maintains the clause-evidence trace relation

$$Trace(\kappa) = evidence(\kappa), \tag{16}$$

and the anchor coverage relation

$$Cover(a_i) = \{\kappa \mid anchor(\kappa) = a_i\}. \tag{17}$$

Missing coverage of explicit anchors forms pending contract content. When pending content affects mission consequences, the system enters the clarification process.

3.3. Contract Admission Under Proof-Obligation Gating

The second class of UAV admission risk comes from task specifications that are formally complete but governance-invalid. A task clause may have readable semantics but lack evidence, exceed source authority, modify non-negotiable constraints, or contain LLM-added assumptions. The proof-obligation gate converts these risks into a unified contract-admission predicate.

EAMSR defines candidate admission in three stages. The governance gate checks evidence, authority, mutability, and untrusted-semantic-isolation obligations:

$$GovOK(C_i) = PO_{\text{E}}(C_i) \wedge PO_{\text{A}}(C_i) \wedge PO_{\text{M}}(C_i) \wedge PO_{\text{U}}(C_i). \tag{18}$$

Candidates that pass governance form the governance-admissible set:

$$\mathcal{C}_{\text{gov}} = \{C_i \in \mathcal{C} \mid GovOK(C_i) = 1\}. \tag{19}$$

If $\mathcal{C}_{\text{gov}} \neq \varnothing$, the system computes consequence signatures and screens pairwise compatibility at the set level using the Consistent predicate (defined in Equation (37)). If the set is consistent, a candidate C^* is selected; the final admission requires a backend witness:

$$\text{AdmissionReady}(C^*, \mathcal{C}_{\text{gov}}) = GovOK(C^*) \wedge \text{Consistent}(\mathcal{C}_{\text{gov}}). \tag{20}$$

This order avoids circularity: consequence screening is evaluated only on governance-admissible candidates. The backend witness PO_{B} (Equation (25)) is then checked separately; an ADMIT decision requires both AdmissionReady and a concrete witness π (Equation (48)).

The evidence obligation PO_{E} requires all hard mission obligations to be supported by admissible evidence or a deterministic trusted derivation:

$$PO_{\text{E}}(C_i) = 1 \iff \forall \kappa \in H_i,\ Sup(\kappa) = \text{Y}. \tag{21}$$

The authority obligation PO_{A} requires source, authority level, and modal strength to be consistent:

$$PO_{\text{A}}(C_i) = 1 \iff \forall \kappa \in H_i \cup S_i,\ AuthOK(\kappa, \Gamma) = 1. \tag{22}$$

The governance policy Γ defines a hierarchy of authority levels (Table 6). When two sources at different authority levels conflict, the higher-authority source prevails. Sources at the same authority level are resolved by recency (more recent prevails) or, if timestamps are equal, by the governance-specified tie-breaking rule. An operator can override a system default at the operational level, and an emergency rule can override a normal geofence policy, but no source can override a regulatory constraint. *Emergency* is not a standing authority level in Table 6; it is a context-dependent override condition activated by a declared contingency (e.g., imminent hazard, medical urgency) that carries the authority of the declaring source (typically operator-level) and is recorded with its triggering condition in the audit trail. An emergency override does not create a new clause authority level; it authorizes a temporary relaxation of specific non-regulatory constraints (geofence buffers, time windows, communication requirements) for the duration of the declared contingency, after which the standard hierarchy resumes. Regulatory constraints remain non-overridable even under emergency conditions.

Table 6. Authority levels in the governance policy Γ.

Level	Description	Overridable by
regulatory	Airspace regulations, no-fly zones, legal mandates	None
operator	Mission commander's explicit authorization	Policy-activated emergency override
operational	Organization-level mission parameters	Operator
system_default	EAMSR default values (e.g., return reserve)	Operational
llm_suggestion	LLM-proposed parameter or constraint	Any of the above

The mutability obligation PO_{M} requires every clause edit to belong to the transformation relation allowed by the governance policy:

$$PO_{\mathrm{M}}(C_i) = 1 \iff \forall e \in Edit(C_i),\ e \in \mathcal{R}_{\Gamma}. \tag{23}$$

The transformation relation $\mathcal{R}_{\Gamma}$ defines the allowed modifications for each clause based on its mutability classification (Table 7). The mutability field interacts with the source field: clauses with source *user_explicit* and mutability *immutable* can only be kept; clauses with source *llm_generated* and mutability *revocable* can be weakened, dropped, or replaced. The refinement relaxation priority is specified by the governance policy Γ; the evaluated configuration relaxes constraints in the order temporal first, then spatial, then communication, then payload, so that operationally critical spatial and payload constraints are preserved as long as possible. Different mission types may benefit from different priority orderings (e.g., emergency delivery may prioritize temporal relaxation, while inspection may prioritize spatial preservation); the framework supports such configuration through Γ.

Table 7. Mutability modes and allowed transformations.

Mutability	Allowed Transformations	Typical Source
immutable	keep only	user_explicit, regulatory
protected	keep, clarify (with authorization)	user_implied
governance	keep, weaken, default (per Γ)	trusted_context
overridable	keep, weaken, drop, default	system_default
revocable	keep, weaken, drop, default, replace	llm_generated

The untrusted-semantic-isolation obligation PO_{U} requires hard mission obligations to contain no unverified hypothetical claims:

$$PO_{\mathrm{U}}(C_i) = 1 \iff Claims(H_i) \cap Claim_{\mathrm{H}} = \varnothing, \tag{24}$$

where $Claims(H_i)$ denotes the set of atomic claims extracted from the hard clause set H_i, and $Claim_{\mathrm{H}}$ is the set of LLM-introduced hypothetical claims without external evidence.

The backend executability obligation PO_{B} requires the existence of a task-level plan witness:

$$PO_{\mathrm{B}}(C) = 1 \iff \exists \pi : Witness(C, \pi) = 1, \tag{25}$$

where the *Witness* predicate is defined in Equation (39).

To illustrate the authority and mutability model, consider a powerline-inspection mission in controlled airspace. The regulatory constraint "no flight above 120 m in Class C airspace" is immutable at the regulatory level: no operator or system default can modify it. The operator's requirement to "inspect towers A through D" is protected at the operator level: the set of inspection targets can be clarified (e.g., if tower C is in a temporary restricted area) but the core inspection objective must be preserved. For the return reserve, we distinguish the non-overridable regulatory lower bound $r^{\mathrm{reg}}_{\mathrm{min}}$ from the operational system default $r^{\mathrm{sys}}_{\mathrm{default}}$, which satisfies $r^{\mathrm{sys}}_{\mathrm{default}} \geq r^{\mathrm{reg}}_{\mathrm{min}}$. The system default $r^{\mathrm{sys}}_{\mathrm{default}} = 15\%$ is overridable at the operational level: an operator may increase it to 20% for conservative missions but cannot decrease it below $r^{\mathrm{reg}}_{\mathrm{min}}$. An LLM-suggested clause "use thermal camera for night inspection" is revocable: if no thermal camera is available in the payload, this clause is dropped without affecting the inspection objective. When the LLM proposes reducing the return reserve to 5% to extend inspection time, this proposal is rejected because the policy specifies a protected minimum $r^{\mathrm{reg}}_{\mathrm{min}} > 5\%$ (an immutable constraint at the regulatory level), enforced by PO_{M}, and because the LLM-suggestion source lacks the authority to modify any return-reserve parameter, enforced by PO_{A}.

3.4. Authority-Bounded Contract Refinement and Mission-Consequence Clarification

The third class of UAV admission risk arises from automatic repair of infeasible tasks. Executability conflicts may result from overly tight time windows, restricted target regions, insufficient communication coverage, payload mismatch, or insufficient return reserve. A refinement mechanism must restore executability while preserving protected invariants, user hard intent, and evidence boundaries.

When direct admission fails, the backend diagnosis interface returns a conflict core:

$$\mathcal{Q} = \mathrm{Diagnose}(C_i, \Omega_{\mathrm{UAV}}), \tag{26}$$

where $\mathcal{Q}$ contains the task clauses, protected invariants, and backend constraints responsible for unsatisfiability. The governance layer converts $\mathcal{Q}$ into a structured diagnostic feedback package:

$$D_i = \langle \mathcal{Q}, failure, allowed, forbidden, required_evidence, affected_consequence \rangle. \tag{27}$$

Under the constraint of D_i, the LLM proposes a repair suggestion:

$$p = \langle \Delta G, Claim(p), action, evidencePtr, expectedConsequence \rangle. \tag{28}$$

In this proposal, ΔG is the mission-obligation difference; *action* is one of weaken, drop, default, split, mark-pending, or clarify; and *expectedConsequence* declares the expected change in mission consequences. Mission consequences are defined by the EAMSR consequence-signature function.

We define the authority-admissible refinement relation as

$$C' \preceq^{\mathrm{adm}}_{\Gamma} C. \tag{29}$$

This relation holds if and only if the following criteria hold jointly:

$$Edit(C, C') \subseteq \mathcal{R}_\Gamma, \tag{30}$$

$$Preserve(I_{\text{prot}}) \wedge Preserve(G_{\text{user-hard}}) \wedge NoHypotheticalAdmission(C') \wedge GovOK(C') = 1. \tag{31}$$

In this constraint, $Preserve(I_{\text{prot}})$ indicates preservation of protected operational invariants, and $Preserve(G_{\text{user-hard}})$ indicates preservation of user hard intent. The term $NoHypotheticalAdmission(C')$ indicates that unverified assumptions do not enter hard mission obligations.

Among candidates satisfying $C' \preceq_\Gamma^{\text{adm}} C$, EAMSR selects the minimum-cost governance-admissible refinement candidate:

$$C^* = \arg\min_{C'} \left[\lambda_{\text{sem}} \tilde{d}_{\text{sem}} + \lambda_{\text{edit}} \widetilde{|Edit|} + \lambda_{\text{pref}} \tilde{d}_{\text{pref}} + \lambda_{\text{plan}} \widetilde{\hat{J}}_{\text{plan}}(C') \right]. \tag{32}$$

In this objective, d_{sem} is semantic displacement, $|Edit|$ is the number of edits, d_{pref} is soft-preference loss, and $\hat{J}_{\text{plan}}(C')$ is the estimated mission-energy cost produced by BackendDiag(C'). Note that $\hat{J}_{\text{plan}}$ is a lightweight diagnostic estimate of the energy required to execute the candidate's mission profile, not the cost of the final witness π, which is generated only after C^* is selected. This separation avoids a circular dependency in which candidate selection would require a witness that is only produced after selection. Because these terms have different native scales, each is normalized to $[0, 1]$ before weighting:

$$\widetilde{|Edit|} = \frac{|Edit|}{B_r}, \qquad \widetilde{\hat{J}}_{\text{plan}}(C') = \min\left(1, \frac{\hat{J}_{\text{plan}}(C')}{J_{\text{ref}}}\right), \tag{33}$$

where B_r is the refinement budget and J_{ref} is a reference diagnostic cost (the battery-energy capacity in the evaluated configuration). The displacement and preference distances $d_{\text{sem}}, d_{\text{pref}} \in [0, 1]$ require no additional normalization. The coefficients satisfy $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, so that each normalized term contributes a bounded share of the total cost. The evaluated configuration uses the uniform profile $\lambda_{\text{sem}} = \lambda_{\text{edit}} = \lambda_{\text{pref}} = \lambda_{\text{plan}} = 0.25$; non-uniform profiles can be configured when a governance policy prioritizes, for example, minimal semantic displacement over plan cost.

Natural-language ambiguity in UAV tasks is screened through a mission-consequence signature. For a candidate contract C, define

$$\chi(C) = \langle feas, goals, optional, order, rth, window, airspace, comm, payload, weather \rangle. \tag{34}$$

The components of $\chi(C)$ are as follows: $feas$ denotes executability; $goals$ denotes the set of hard goals; $optional$ denotes the set of optional tasks; $order$ denotes the required order closure; rth denotes the minimum return-energy margin; $window$ denotes the minimum time-window margin; $airspace$ denotes airspace and geofence compliance; $comm$ denotes communication and data-return feasibility; $payload$ denotes payload and sensor satisfaction; and $weather$ denotes weather and observation-condition satisfaction.

Each component of $\chi(C)$ is normalized to $[0, 1]$ so that heterogeneous Boolean, categorical, and numerical quantities can be combined, and the component distances are defined per component family as summarized in Table 8. The mission-consequence distance between two contracts is the weighted average of the component distances:

$$\Delta_\chi(C_i, C_j) = dist(\chi(C_i), \chi(C_j)) = \frac{\sum_{k=1}^{10} w_k \, d_k(\chi(C_i), \chi(C_j))}{\sum_{k=1}^{10} w_k}, \qquad w_k \geq 0. \tag{35}$$

The normalization keeps $\Delta_\chi \in [0,1]$, so that the compatibility threshold ϵ_χ has a uniform interpretation across mission types. The evaluated configuration uses uniform weights $w_k = 1$ for $k = 1, \ldots, 10$.

The backend provides two interfaces. BackendDiag(C) is a lightweight task-level diagnostic evaluated for all governance-admissible candidates; it produces satisfaction ratios for feasibility, airspace compliance, communication coverage, payload, and weather without constructing a full action sequence. These ratios feed the consequence signature $\chi(C)$ and enable mission-consequence screening (MCS) before candidate selection. The final backend witness $Witness(C^*, \pi)$ is generated only for the selected candidate C^*; it constructs a complete action sequence π and verifies SafeRTH, GeoSafe, and AuxOK on the full plan.

Table 8. Component distances of the mission-consequence signature $\chi(C)$. All components are normalized to $[0,1]$ before combination.

Components	Normalization	Component Distance
$feas$, $airspace$, $comm$, $payload$, $weather$	satisfaction ratio in $[0,1]$ produced by BackendDiag (lightweight task-level diagnostic, distinct from the final witness)	absolute difference
$goals$, $optional$, $order$	sets over the task-ontology vocabulary; for $order$, the set of precedence pairs (g, g')	Jaccard distance $1 - \lvert S_i \cap S_j \rvert / \lvert S_i \cup S_j \rvert$ (0 if both sets are empty)
rth, $window$	truncated normalized margins $\min(1, margin/margin_{\text{ref}})$ and $\min(1, slack/slack_{\text{ref}})$, with $margin_{\text{ref}} = 0.30$ and $slack_{\text{ref}} = 0.50$	absolute difference of the normalized margins

A single global weighted-average threshold can mask large changes in individual critical dimensions: with 10 uniform-weight components, a complete reversal in one dimension ($d_k = 1$) contributes only 0.1 to Δ_χ, which may remain below $\epsilon_\chi = 0.15$. To prevent this, the compatibility relation augments the global threshold with a per-dimension hard veto on critical components. Given threshold ϵ_χ and per-dimension thresholds τ_k, mission-consequence compatibility is defined as

$$C_i \sim_\chi C_j \iff \left[\max_{k \in K_{\text{crit}}} d_k(C_i, C_j) \leq \tau_k\right] \wedge \left[\Delta_\chi(C_i, C_j) \leq \epsilon_\chi\right], \tag{36}$$

where $K_{\text{crit}} = \{goals, feas, airspace, rth\}$ is the set of critical dimensions whose individual changes must not exceed per-dimension thresholds τ_k, and the remaining components (optional preferences, order, communication, payload, weather, window) are evaluated only through the weighted average. In the evaluated configuration, $\tau_k = 0.3$ for all $k \in K_{\text{crit}}$, so that any critical-dimension change exceeding 30% of its normalized range triggers incompatibility regardless of the aggregate distance. The threshold $\tau_k = 0.3$ was selected using the same 20-task development subset used for ϵ_χ and λ: values below 0.2 caused spurious CLARIFY decisions for contracts that differed only in non-critical refinements, while values above 0.4 allowed goal-set reversals that the aggregate distance alone would have caught. The framework admits dimension-specific thresholds or governance-defined compatibility criteria through the weights w_k and per-dimension configuration; calibrating such criteria with operational stakeholders is a direction for future work.

Because each component distance is normalized to $[0,1]$ and Δ_χ is a weighted average, the relation $\sim_\chi$ is reflexive and symmetric. It is generally not transitive—candidates A, B, and C may satisfy the veto and aggregate conditions pairwise for (A,B) and (B,C) while failing for (A,C)—so $\sim_\chi$ is a tolerance relation, not a strict equivalence. To decide

whether a candidate set is consequence-consistent, we require pairwise compatibility under Equation (36):

$$\text{Consistent}(\mathcal{C}) \iff \forall\, C_i, C_j \in \mathcal{C} :\ C_i \sim_\chi C_j. \tag{37}$$

When admissible candidates or refinement candidates do not satisfy pairwise consistency, the system returns CLARIFY. When all admissible candidates are pairwise compatible, the system selects C^* according to semantic displacement, soft-preference satisfaction, and estimated mission-energy cost.

Four examples illustrate consequence compatibility and its violation under the evaluated configuration ($\epsilon_\chi = 0.15$, $\tau_k = 0.3$, uniform weights). (i) Two contracts share all components except the return-energy margin, whose raw values of 0.15 and 0.18 give normalized margins $m_{rth} = 0.50$ and $m_{rth} = 0.60$; the resulting distance is $\Delta_\chi = 0.10/10 = 0.01 \leq \epsilon_\chi$ and no critical-dimension distance exceeds τ_k, so the contracts are consequence-compatible. (ii) Two contracts differ only in the time-window margin ($d_{window} = 0.20$) and payload satisfaction ($d_{payload} = 0.10$), giving $\Delta_\chi = 0.03$; they are likewise compatible, and the refinement that better satisfies soft preferences is selected without clarification. Conversely, clarification is triggered when candidates are not pairwise compatible. (iii) Two candidate contracts with disjoint hard-goal sets ($d_{goals} = 1 > \tau_k = 0.3$) trigger the critical-dimension veto, so the pair is incompatible regardless of the aggregate distance. (iv) A candidate whose feasibility, airspace, and communication components are degraded ($d_{feas} = d_{airspace} = 0.8 > \tau_k = 0.3$) likewise triggers the veto through feas and airspace (both in K_{crit}); the communication degradation ($d_{comm} = 0.8$) does not enter the veto, since comm is not in K_{crit}, and contributes only to the aggregate $\Delta_\chi = 0.24 > \epsilon_\chi$. In both cases, the system returns CLARIFY instead of silently selecting one consequence outcome.

3.5. Backend Plan Witness, Audit Trail, and Method Properties

The final admission risk is a specification that is semantically reasonable but task-inexecutable. The backend witness layer binds the admitted contract to UAV task constraints, so that each ADMIT output carries task-level executability evidence.

Let the plan witness be a finite action sequence:

$$\pi = [(a_1, t_1^s, t_1^e), (a_2, t_2^s, t_2^e), \ldots, (a_m, t_m^s, t_m^e)], \tag{38}$$

where a_j is a task-level action such as takeoff, navigation, inspection, imaging, hovering, data return, delivery, return-to-home, or abort, and t_j^s and t_j^e are the start and end times of the action.

The backend witness condition is

$$Witness(C^*, \pi) = 1 \iff \pi \vDash A_{\text{ctx}} \wedge I_{\text{prot}} \wedge H^* \wedge \Omega_{\text{UAV}}, \tag{39}$$

where $H^* = Hard(G^*)$ is the set of hard goals and hard constraints extracted from the selected contract C^*. Soft preferences S^* influence candidate ranking (Equation (32)) but are not part of the witness condition, and pending clauses P^* are excluded from H^* by the evidence obligation PO_{E}. This paper uses return-to-home safety and airspace safety as core UAV safety constraints, while communication, payload, time-window, and weather constraints are incorporated into Ω_{UAV}.

Return-to-home safety is defined as

$$SafeRTH(\pi) = 1 \iff \forall t,\ E_{\text{eff}}(t) \geq (1+\beta)\, E_{\text{RTH}}(pos(t), t) + E_{\text{res}}, \tag{40}$$

where $\beta = 0.10$ is a safety buffer on the return-to-home energy estimate, $E_{\text{eff}}(t)$ is the effective available energy at time t (the SOC-corrected remaining energy defined in

Equation (42)), $E_{\mathrm{RTH}}(pos(t), t)$ is the minimum energy required to safely return from the current position, and E_{res} is the safety reserve. The return-to-home energy estimate E_{RTH} accounts for horizontal distance to the home position, vertical altitude difference, landing energy, and a 30 s pre-landing hover:

$$E_{\mathrm{RTH}} = e_{\mathrm{horiz}} \cdot d_{\mathrm{horiz}}^{\mathrm{home}} + e_{\mathrm{vert}} \cdot |dz_{\mathrm{home}}| + E_{\mathrm{land}} + E_{\mathrm{hover,30s}}, \tag{41}$$

where $e_{\mathrm{horiz}} = 0.5\,\mathrm{Wh/m}$, $e_{\mathrm{vert}} = 1.0\,\mathrm{Wh/m}$, $E_{\mathrm{land}} = 5\,\mathrm{Wh}$, and $E_{\mathrm{hover,30s}} \approx 1.67\,\mathrm{Wh}$. The safety reserve is $E_{\mathrm{res}} = 0.15 \times 500 = 75\,\mathrm{Wh}$. When the battery state of charge (SOC) falls below 30%, the effective available energy is reduced by a piecewise correction factor $f(SOC)$ that accounts for increased internal resistance at low charge levels:

$$E_{\mathrm{eff}}(t) = f(SOC(t)) \cdot E(t), \quad f(SOC) = \begin{cases} 1.0, & SOC \geq 0.3, \\ 1.0 - 0.5(0.3 - SOC), & SOC < 0.3, \end{cases} \tag{42}$$

so that at $SOC = 0.15$ the correction factor is $f = 0.925$. The normalized energy margin at each time point is $M_{\mathrm{RTH}}(t) = (E_{\mathrm{eff}}(t) - (1+\beta)\, E_{\mathrm{RTH}}(pos(t), t) - E_{\mathrm{res}}) / E_{\mathrm{RTH}}(pos(t), t)$, and the SafeRTH condition in Equation (40) is equivalent to requiring $M_{\mathrm{RTH}}(t) \geq 0$ at all sampled times.

Airspace safety is defined as

$$GeoSafe(\pi) = 1 \iff \forall t,\ pos(t) \notin NFZ \wedge pos(t) \in Zone_{\mathrm{op}}. \tag{43}$$

In this equation, NFZ is the set of no-fly zones, and $Zone_{\mathrm{op}}$ is the permitted operational airspace.

Other task-level constraints are represented by the unified predicate

$$AuxOK(\pi) = DataReturn(\pi) \wedge PayloadOK(\pi) \wedge WindowOK(\pi) \wedge WeatherOK(\pi). \tag{44}$$

The backend witness evaluates the candidate contract against a task-level discrete UAV simulation model. The simulator maintains a state vector at sampled time points along the witness plan, tracking position (local Cartesian coordinates in metres), remaining battery energy, communication coverage ratio, payload status, and elapsed mission time. The witness plan is a finite sequence of task-level actions, where each action type has an associated energy cost model and switch overhead (Table 9).

Table 9. Backend action types and energy model. Switch overhead contributes hover energy during action transitions.

Action	Overhead (s)	Energy Model
takeoff	15	Fixed takeoff energy (10 Wh)
fly	0	Horizontal: $e_{\mathrm{horiz}} \times d_{\mathrm{horiz}}$; vertical: $e_{\mathrm{vert}} \times \lvert dz \rvert$
inspect	5	Hover energy during duration
capture	10	Hover energy during duration
hover	5	$P_{\mathrm{hover}} \times \Delta t / 3600$
transmit	20	Hover energy during duration
deliver	10	Hover energy during duration
return	0	Flight energy to home position
abort	5	Hover energy during duration
landing	5	Fixed landing energy (5 Wh)

Witness-plan generation uses a greedy forward search with bounded backtracking over the task-level action vocabulary, advancing the state vector at a fixed 1 s timestep and checking all safety and auxiliary predicates at each step. The search terminates when a complete feasible plan is found, all branches within the depth limit are exhausted, or the per-contract wall-clock budget of 5 s is reached. The action priority ordering is takeoff → fly (to first target) → inspect/capture/deliver (mission actions in goal order) → transmit (if required) → return → landing, with abort inserted at any point if SafeRTH is violated. Backtracking reorders optional mission actions and adjusts waypoint sequencing within the depth limit $D_{\max} = 20$. The same L1 backend model is used for benchmark label construction and admission decisions; this shared dependency is acknowledged as a limitation in Section 4.6.

The present backend uses a simplified task-level energy proxy: the energy model accounts for horizontal distance, vertical altitude, landing, and hover, but does not model continuous wind- or payload-dependent aerodynamic power consumption. Wind speed is checked as a binary threshold in *WeatherOK*, and payload affects sensor availability rather than energy draw. This simplification is intentional for the task-level admission layer, where the goal is to verify executability at the mission-specification level rather than to produce a flight-controller-grade trajectory energy estimate.

Communication coverage is evaluated using a distance-based attenuation model rather than a binary in/out check. The coverage ratio decreases linearly from 1.0 inside the inner radius ($0.8 \times r_{\max}$) to 0.0 at the outer radius ($r_{\max}$), and the plan passes when the minimum coverage ratio along the flight path is at least 0.5. The auxiliary constraints $AuxOK$ verify that: (i) a data-return action exists when communication is required; (ii) all sensors specified in payload clauses are available; (iii) all actions fall within their designated time windows; and (iv) no action exceeds the maximum wind speed from weather constraints.

The abstract witness condition was defined in Equation (39). The UAV task model Ω_{UAV} is decomposed into implementation-level predicates:

$$\Omega_{\text{UAV}} = SafeRTH \wedge GeoSafe \wedge AuxOK \wedge \Omega_{\text{task}}, \tag{45}$$

where Ω_{task} captures scenario-specific constraints not covered by the three standard predicates. Substituting Equation (45) into Equation (39) yields the implementation-level witness used in the evaluation:

$$\begin{aligned} Witness(C^*, \pi) = SafeRTH(\pi) \wedge GeoSafe(\pi) \wedge AuxOK(\pi) \\ \wedge \left(\pi \vDash A_{\text{ctx}} \wedge I_{\text{prot}} \wedge H^* \wedge \Omega_{\text{task}}\right). \end{aligned} \tag{46}$$

A failed witness attempt does not by itself establish infeasibility: the search is bounded by $D_{\max}$ and a wall-clock budget, so termination without a witness does not prove that no witness exists. Contract-level infeasibility is reported only when the backend returns a conclusive infeasibility status (Equation (47)); otherwise, failure to construct a witness remains inconclusive. The witness verification collects all violations without early exit, producing a structured conflict-core diagnosis that identifies the specific clauses, protected invariants, and backend constraints responsible for unsatisfiability. Verification completes in under 5 s per contract on a single core.

Because witness search is bounded, the backend therefore reports a three-valued status:

$$\text{BackendStatus}(C) \in \{\textsc{WitnessFound}, \textsc{ConclusiveInfeasible}, \textsc{Inconclusive}\}. \tag{47}$$

WitnessFound means that a concrete witness π satisfying Equation (46) has been constructed; it can support an ADMIT decision. ConclusiveInfeasible means that unsatisfiability is conclusively established under the declared backend model: the diagno-

sis identifies a deterministic conflict with an immutable or protected constraint that no authority-bounded refinement can remove (e.g., a hard energy deficit against battery capacity, or an airspace breach of a non-overridable no-fly zone), or the search has exhaustively covered the finite task-level action space declared for the scenario without finding a witness. Only CONCLUSIVEINFEASIBLE can support a REJECT decision. INCONCLUSIVE covers the remaining outcomes—wall-clock timeout, exhaustion of the depth or backtracking budget, or termination without a witness and without a conclusive infeasibility certificate. INCONCLUSIVE is not treated as proof of infeasibility: the blocking condition is considered unresolved and the request is returned as CLARIFY, since a refined contract, additional operator input, or a larger search budget might still yield a witness.

The final decision rule is

$$decision = \begin{cases} \text{ADMIT}, & \exists C^*, \pi: \ \text{AdmissionReady}(C^*, \mathcal{C}_{\text{gov}}) = 1 \wedge Witness(C^*, \pi) = 1, \\ \text{REJECT}, & \text{an immutable or protected admission condition is violated, or task-level} \\ & \text{infeasibility is conclusively established by a backend diagnosis} \\ & (\text{BackendStatus}(C^*) = \text{CONCLUSIVEINFEASIBLE}), \\ \text{CLARIFY}, & \text{the blocking condition is remediable through additional evidence, authorized} \\ & \text{modification, or resolution of consequence-sensitive ambiguity, or backend} \\ & \text{closure remains inconclusive.} \end{cases} \tag{48}$$

REJECT asserts that no admissible resolution exists within the currently declared trusted context, governance policy, protected constraints, admissible refinement space, and conclusively evaluated backend conditions; it does not assert that the mission can never become executable under any future evidence, authorization, or refinement. Failure to obtain a witness solely because of search-depth or wall-clock limits is treated as unresolved (INCONCLUSIVE) and leads to CLARIFY rather than REJECT, preserving the conservative direction of the admission boundary.

EAMSR outputs the audit trail

$$\begin{aligned} \mathcal{T}_{\text{audit}} = \langle & \mathcal{A}, \mathcal{K}, Trace, Cover, Authority, Mutability, \\ & UntrustedClaims, PO, \mathcal{Q}, Edit, \chi, \pi, decision \rangle. \end{aligned} \tag{49}$$

This trail connects natural-language anchors, candidate clauses, evidence sources, anchor coverage, authority states, mutability states, untrusted semantic increments, proof-obligation results, conflict cores, refinement edits, mission-consequence signatures, backend plan witnesses, and the final admission decision.

The following properties are specification-level properties of the abstract admission model. They hold provided that clause extraction, evidence status, authority assignments, governance policy, and backend predicates are implemented consistently with the formal definitions; they do not constitute machine-verified guarantees of the complete operational system.

Property 1: Protected-invariant preservation. If $c \in I_{\text{prot}}$ and Γ marks c as immutable, then every ADMIT output preserves c:

$$c \in I_{\text{prot}} \Rightarrow Preserve(c, G^*). \tag{50}$$

This holds because the mutability proof obligation PO_{M} and $Preserve(I_{\text{prot}})$ jointly exclude any candidate that modifies, weakens, or removes protected invariants from the admissible refinement set.

Property 2: Represented user hard intent cannot be silently modified. Given correct anchor extraction, if a user hard intent $g \in G_{\text{user-hard}}$ conflicts with protected constraints or the backend model, the system returns CLARIFY or REJECT:

$$g \in G_{\text{user-hard}} \wedge \neg\exists \pi : \pi \models g \wedge I_{\text{prot}} \wedge \Omega_{\text{UAV}} \Rightarrow \mathit{decision} \in \{\text{CLARIFY}, \text{REJECT}\}. \quad (51)$$

The preservation constraint and the mission-consequence clarification rule jointly guarantee this property. The qualifier "represented" is necessary because anchor extraction is treated as trusted preprocessing (Section 3.2); if the extractor omits a user hard intent, the governance layer has no opportunity to protect it.

Property 3: Unverified assumptions do not propagate. If an atomic claim ϕ belongs to $\mathit{Claim}_{\text{H}}$ and cannot be verified by $\mathcal{E}$, Γ, or Ω_{UAV}, then ϕ does not enter any admitted hard mission obligation:

$$\phi \in \mathit{Claim}_{\text{H}} \wedge (\mathcal{E} \cup \Gamma \cup \Omega_{\text{UAV}}) \not\models \phi \Rightarrow \phi \notin \mathit{Claims}(\mathit{Hard}(G^*)). \quad (52)$$

This is a direct consequence of the untrusted-isolation proof obligation PO_{U}. Unverified semantic increments remain in pending items, clarification requests, or the audit trail.

Property 4: Bounded closure. Algorithm 1 performs at most B_c candidate-generation iterations (counter b) followed by at most B_r refinement iterations (counter r), with duplicate candidates prevented from resubmission. The total number of LLM invocations is therefore bounded by

$$T \leq B_c + B_r. \quad (53)$$

This property establishes EAMSR as a bounded admission-closure process.

Assumptions and guarantee boundary. Properties 1–4 are conditional on the five explicit assumptions in Table 10. Under A1–A5, an ADMIT decision enforces the specified admission predicate *by construction*; the properties do not constitute machine-verified guarantees of the complete operational system, and A5 makes explicit that a missing witness is never interpreted as proof of infeasibility unless the backend conclusively establishes it (Equation (47)).

Table 10. Assumptions underlying the specification-level properties (Properties 1–4).

ID	Assumption	Content
A1	Anchor representation	Validated anchors correctly represent the user hard intent relevant to admission (Property 2).
A2	Evidence fidelity	Admitted evidence sources are correct, current, and correctly prioritized.
A3	Governance fidelity	Authority and mutability policies correctly encode the intended operational rules.
A4	Backend fidelity	Ω_{UAV} is an adequate task-level abstraction for the evaluated admission decision.
A5	Witness semantics	A positive witness establishes model-level executability; absence of a witness is not treated as proof of infeasibility unless infeasibility is conclusively established under the declared backend model (Equation (47)).

Algorithm 1 EAMSR mission admission via evidence-carrying MAC.

Require: Instruction x, context A_{ctx}, invariants I_{prot}, policy Γ, evidence $\mathcal{E}$, backend Ω_{UAV}, candidate budget B_c, refinement budget B_r
Ensure: *decision*, G^*, π, $\mathcal{T}_{\text{audit}}$

Extract anchors $\mathcal{A}$ from x.
Initialize $\mathcal{C} \leftarrow \varnothing$, $b \leftarrow 0$, $r \leftarrow 0$, *decision* $\leftarrow$ UNSET, $G^* \leftarrow \bot$, $\pi \leftarrow \bot$.
while $b < B_c$ **do**
  Generate response and parse clause set $\mathcal{K}^{(b)}$ via Equation (7).
  if the response is parseable **then**
    Assemble candidate C_{b+1}.
    if C_{b+1} is not a duplicate of any existing candidate **then**
      Add C_{b+1} to $\mathcal{C}$.
    end if
  else
    Record PARSEFAILURE in $\mathcal{T}_{\text{audit}}$.
  end if
  $b \leftarrow b + 1$.
end while
if $\mathcal{C} = \varnothing$ **then**
  decision $\leftarrow$ CLARIFY; assemble $\mathcal{T}_{\text{audit}}$ and
  return *decision*, G^*, π, $\mathcal{T}_{\text{audit}}$.
end if
while *decision* $=$ UNSET **and** $r \leq B_r$ **do**
  Apply *GovOK* to obtain $\mathcal{C}_{\text{gov}}$.
  if $\mathcal{C}_{\text{gov}} = \varnothing$ **then**
    $\mathcal{Q} \leftarrow \text{Diagnose}(\mathcal{C}, \Gamma)$.
    if $\mathcal{Q}$ is governance-remediable **and** $r < B_r$ **then**
      $\mathcal{C} \leftarrow \text{Refine}(\mathcal{C}, \mathcal{Q}, \Gamma)$; $r \leftarrow r + 1$.
    else
      decision $\leftarrow$ REJECT if $\mathcal{Q}$ is an immutable/protected-requirement conflict, else CLARIFY.
    end if
  else
    Compute χ; screen Consistent($\mathcal{C}_{\text{gov}}$).
    if Consistent($\mathcal{C}_{\text{gov}}$) **then**
      Select C^* via Equation (32); run the bounded witness search for C^* (Equation (47)).
      if BackendStatus(C^*) $=$ WITNESSFOUND with witness π **then**
        decision $\leftarrow$ ADMIT; $G^* \leftarrow G(C^*)$.
      else
        $\mathcal{Q} \leftarrow \text{Diagnose}(C^*, \Omega_{\text{UAV}})$.
        if $\mathcal{Q}$ is backend-remediable **and** $r < B_r$ **then**
          $\mathcal{C} \leftarrow \text{Refine}(C^*, \mathcal{Q}, \Gamma)$; $r \leftarrow r + 1$.
        else if BackendStatus(C^*) $=$ CONCLUSIVEINFEASIBLE **then**
          decision $\leftarrow$ REJECT.
        else
          decision $\leftarrow$ CLARIFY (INCONCLUSIVE or operator input required).
        end if
      end if
    else
      $\mathcal{Q} \leftarrow \text{Diagnose}(\mathcal{C}_{\text{gov}})$.
      if $\mathcal{Q}$ is consequence-remediable **and** $r < B_r$ **then**
        $\mathcal{C} \leftarrow \text{Refine}(\mathcal{C}_{\text{gov}}, \mathcal{Q}, \Gamma)$; $r \leftarrow r + 1$.
      else
        decision $\leftarrow$ CLARIFY.
      end if
    end if
  end if
end while
if *decision* $=$ UNSET **then**
  decision $\leftarrow$ CLARIFY.
end if
Assemble $\mathcal{T}_{\text{audit}}$ and
return *decision*, G^*, π, $\mathcal{T}_{\text{audit}}$.

4. Experiments and Evaluation

The experiments address three questions: whether joint admission reduces unwarranted admission; whether authority and mission-consequence constraints prevent feasibility from being recovered through semantic migration; and whether backend witnesses close admission decisions with respect to UAV task-level constraints. The evaluation boundary

lies between task submission and task planning, and does not cover continuous trajectory control, flight-control stability, or runtime obstacle avoidance.

4.1. Evaluation Protocol

The evaluation protocol comprises the benchmark and human labels (EAMSR-Bench), the controlled comparison configurations and their coverage boundary, the metric definitions with uncertainty intervals, and the implementation settings.

4.1.1. EAMSR-Bench and Human Labels

EAMSR-Bench is a controlled, internally constructed architecture-stress benchmark. It contains 120 natural-language UAV mission instructions, crossing six mission scenarios with six admission-risk types. Each sample includes a task instruction, trusted context, protected operational constraints, evidence base, governance policy, task-level backend model, and human admission label. The crossed design places multiple risk types in each scenario, so that conclusions do not depend on a single task type.

Instruction generation. The 120 instructions are in Chinese, authored by the research team from Chinese-language operator manuals, civil-aviation regulations, and mission-planning guidelines. No instructions were copied from operational flight logs or obtained from external UAV operators. The six mission scenarios—powerline inspection, disaster-area search, campus delivery, river monitoring, bridge inspection, and communication-limited tasks—were selected to cover the principal UAV mission categories identified in the operational literature. The six admission-risk types were defined by the research team before benchmark construction, derived from the proof-obligation structure of the EAMSR framework. The benchmark was developed concurrently with the framework: risk-type definitions were motivated by the proof obligations, but the specific instructions, evidence bases, and governance policies were authored independently of the EAMSR implementation code. This concurrent development creates a risk of circular validation, which is acknowledged in Section 4.6; the results should therefore be interpreted as performance under a controlled, internally designed benchmark. English renderings in this paper (e.g., Table 11 and Figures 2–7) are glosses of the Chinese originals.

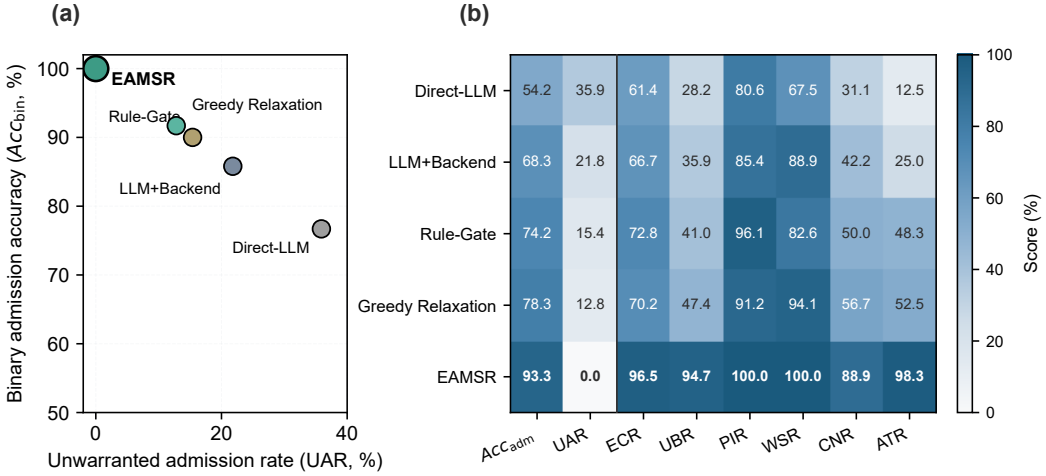


Figure 2. Overall admission performance under the shared task-level protocol. (**a**) Binary admission accuracy (Acc_{bin}) versus unwarranted admission rate for EAMSR and four controlled comparison configurations. (**b**) Heatmap of primary, secondary, and diagnostic metrics; lower values are better for UAR, whereas higher values are better for all other metrics.

Table 11. Overall admission performance. Primary-tier metrics (Acc_{bin}, UAR) are binary ADMIT/non-ADMIT and fair across all configurations. Secondary-tier metric (Acc) is three-class accuracy, post hoc mapped for non-EAMSR configurations. Bold values indicate the best result in each column.

Configuration	Primary (Binary)		Secondary/Diagnostic						
	Acc_{bin}	UAR	Acc	ECR	UBR	PIR	WSR	CNR	ATR
Direct-LLM	76.7	35.9	54.2	61.4	28.2	80.6	67.5	31.1	12.5
LLM+Backend	85.8	21.8	68.3	66.7	35.9	85.4	88.9	42.2	25.0
Rule-Gate	90.0	15.4	74.2	72.8	41.0	96.1	82.6	50.0	48.3
Greedy Relaxation	91.7	12.8	78.3	70.2	47.4	91.2	94.1	56.7	52.5
EAMSR	**100.0**	**0.0**	**93.3**	**96.5**	**94.7**	**100.0**	**100.0**	**88.9**	**98.3**

Note: All values are percentages. Acc_{bin} = binary admission correctness (ADMIT/non-ADMIT agreement with human labels); UAR = unwarranted admission rate (unwarranted ADMIT / 78 non-admissible tasks); Acc = three-class admission accuracy; ECR = evidence coverage rate; UBR = USI blocking rate; PIR = protected-invariant preservation rate; WSR = witness success rate; CNR = consequence clarification rate; ATR = audit-trail completeness rate. $Acc_{\text{bin}} \geq Acc$ because CLARIFY↔REJECT errors do not affect binary correctness. Instruction entries are English glosses of the Chinese originals; full bilingual records are in the reproducibility package.

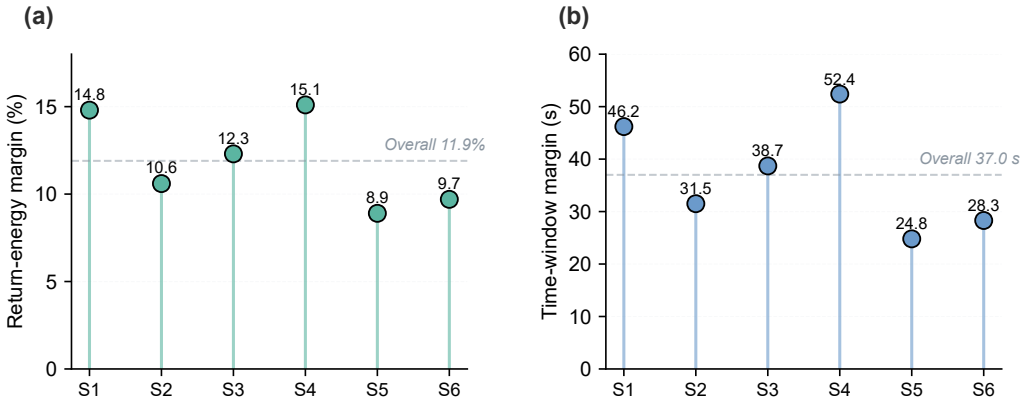


Figure 3. Backend-witness margins across UAV scenarios. (**a**) Minimum return-energy margin for admitted tasks. (**b**) Minimum time-window margin for the same tasks. Dashed horizontal lines denote the overall mean margin across scenarios; all scenario-level minima remain above the zero-margin feasibility boundary.

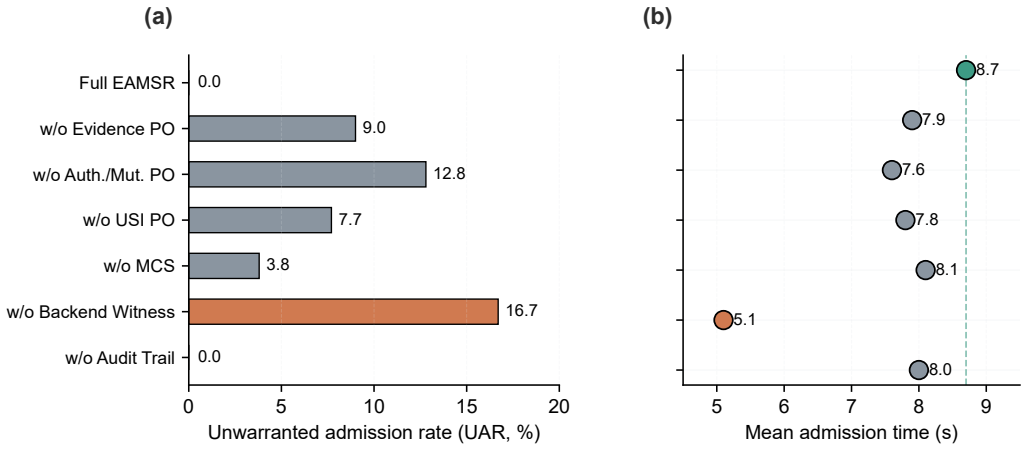


Figure 4. Ablation-induced unwarranted admission and runtime overhead. (**a**) UAR for the full system and six ablated variants. (**b**) Mean admission time under the same protocol; the dashed vertical line denotes the Full EAMSR mean admission time (8.7 s). Removing the backend witness reduces latency most strongly but causes the largest increase in direct admission of inadmissible tasks. (BW-Operator is the audit field passing governance-layer authorization status to the backend interface, not an independent backend proof obligation.)

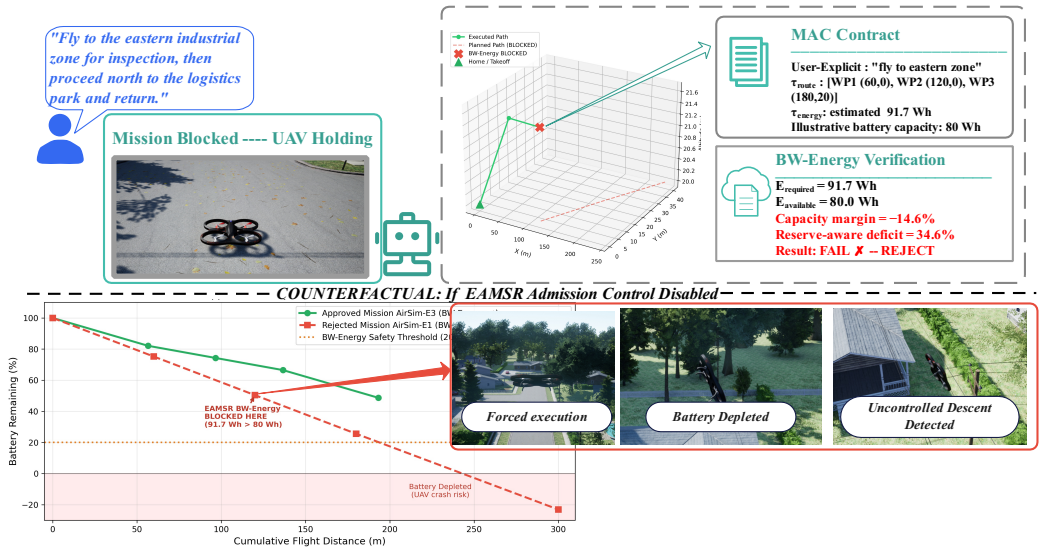


Figure 5. AirSim-E1 energy-infeasible mission and counterfactual execution. The candidate mission requires 91.7 Wh, exceeding both the 80.0 Wh battery capacity and the 64.0 Wh mission-usable energy after reserving 20% for return. EAMSR rejects the mission before takeoff; forced execution enters the unsafe depletion region during return.

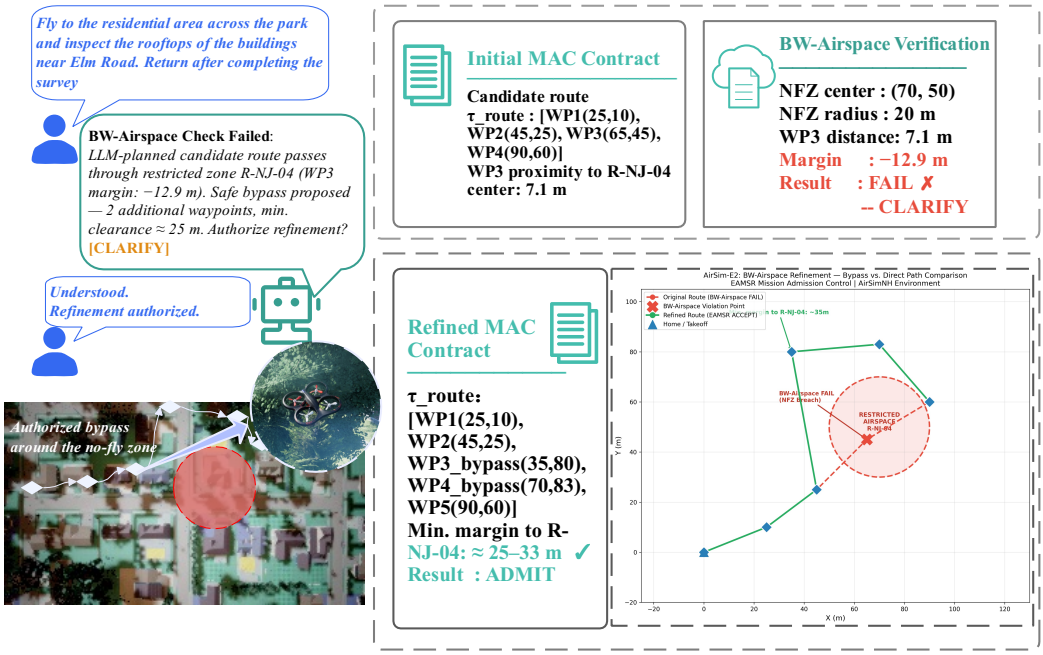


Figure 6. AirSim-E2 restricted-airspace conflict and authorized refinement. The initial route violates the 20 m restricted-airspace buffer and is not admitted. EAMSR requests operator authorization; after authorization, a downstream planner produces a bypass route whose task-level witness satisfies the airspace constraint.

The six admission-risk types define the admission challenge of each task. T1 (normal-risk) tasks are well-specified, evidence-sufficient, and feasible under the given context and backend model, so that all proof obligations are satisfied. T2 (evidence-insufficient) tasks lack evidential support for key parameters or conditions. T3 (authority-conflicting) tasks request actions that conflict with authority rules, governance policy, or mutability constraints. T4 (LLM-assumption contamination) tasks are susceptible to unsupported assumptions introduced during semantic interpretation or repair. T5 (backend-infeasible) tasks appear reasonable at the language level but violate backend constraints on energy, airspace, ge-

ofence, payload, communication, or time windows. T6 (consequence-ambiguous) tasks admit multiple possible mission-consequence profiles, so that the correct outcome depends on resolving consequence-profile ambiguity during mission-consequence screening. Each task is generated by instantiating one scenario with one risk type: scenario-specific mission content (objectives, target areas, and operating conditions) is combined with a risk-type template that injects the corresponding proof-obligation failure or consequence ambiguity into the instruction, evidence base, or context, while T1 templates leave all obligations satisfiable. The resulting 120 tasks contain 30 T1 and 18 tasks of each of T2–T6 per the crossed design (Table 12). T1 is assigned more samples than individual risk types because it is the admissible baseline and provides a stronger assessment of whether the added governance layers cause excessive false clarification or rejection of otherwise admissible missions.

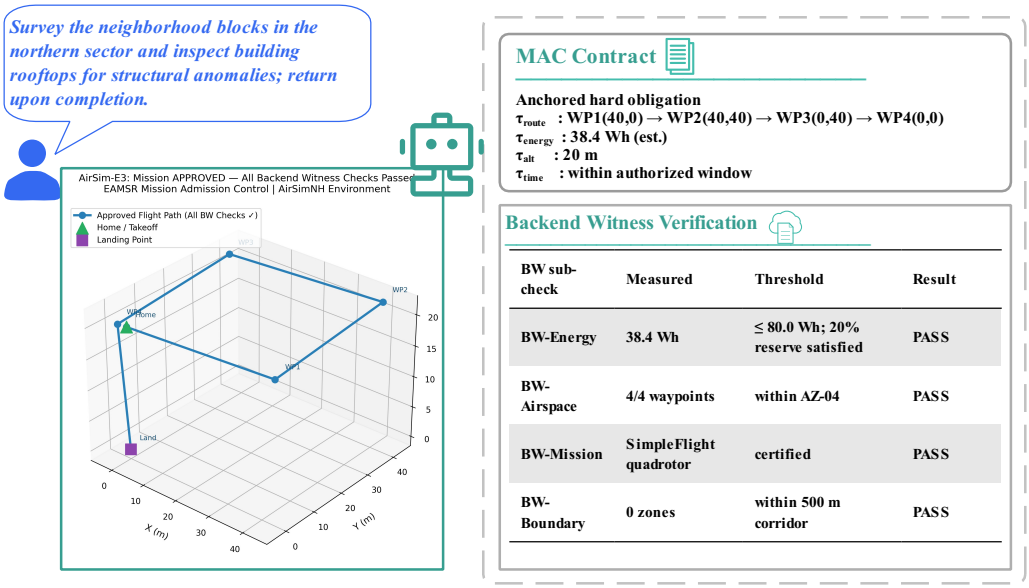


Figure 7. AirSim-E3 admitted grid-inspection mission with backend-witness closure. The mission satisfies evidence, authority, consequence-consistency, and task-level backend conditions. Its estimated energy use is 38.4 Wh, leaving 52.0% battery and a 32.0-percentage-point margin above the return-reserve threshold.

Table 12. EAMSR-Bench composition.

Scenario	T1	T2	T3	T4	T5	T6	Total
S1 Powerline inspection	5	3	3	3	3	3	20
S2 Disaster-area search	5	3	3	3	3	3	20
S3 Campus delivery	5	3	3	3	3	3	20
S4 River monitoring	5	3	3	3	3	3	20
S5 Bridge inspection	5	3	3	3	3	3	20
S6 Communication-limited task	5	3	3	3	3	3	20
Total	30	18	18	18	18	18	120

The human labels include 42 ADMIT, 47 CLARIFY, and 31 REJECT samples (Table 13). ADMIT means that the task can enter the planning chain under the given context and governance policy. CLARIFY means that evidence must be added, authorization must be obtained, or a consequence-profile ambiguity must be resolved. REJECT means that no admissible resolution exists within the declared trusted context, governance policy, protected constraints, and conclusively evaluated backend conditions.

Table 13. Reference label distribution by mission scenario.

Scenario	ADMIT	CLARIFY	REJECT	Total
S1 Powerline inspection	7	8	5	20
S2 Disaster-area search	7	8	5	20
S3 Campus delivery	7	7	6	20
S4 River monitoring	7	8	5	20
S5 Bridge inspection	7	8	5	20
S6 Communication-limited	7	8	5	20
Total	42	47	31	120

Human labels were assigned through a blinded dual-annotation protocol by non-developer domain experts. Each sample was annotated independently by two laboratory domain experts, each with over three years of UAV mission-planning and autonomous-system research experience, who received identical raw inputs (the instruction, trusted context, protected invariants, evidence base, governance policy, and backend model) and were blind to all system outputs, including EAMSR predictions and audit trails. The annotators were not involved in designing the EAMSR admission logic or the benchmark risk-type taxonomy; they received the benchmark samples and a labeling protocol but no information about the framework's proof-obligation structure or decision rules. This separation mitigates, though does not fully eliminate, the risk that the labels encode the same admission logic as the framework, because the benchmark instructions and risk-type definitions were authored by the research team that also developed EAMSR (Section 4.1.1). Annotators followed a fixed procedure that checks, in turn, evidence sufficiency, authority and governance compliance, LLM-assumption risk, backend feasibility, and consequence ambiguity before assigning a label. The boundary between CLARIFY and REJECT was operationalized by a remediability criterion: if the gap can be filled by adding evidence, obtaining authorization, or resolving a consequence-profile ambiguity, the label is CLARIFY; if the request conflicts with an immutable or protected admission condition or is conclusively infeasible under the declared backend model, so that no admissible resolution exists within the declared trusted context, governance policy, protected constraints, and admissible refinement space, the label is REJECT. Disagreements were resolved by a senior annotator whose decision was binding, with escalation to a domain-expert panel majority vote for novel cases. Inter-annotator agreement was measured by pairwise Cohen's kappa over the three labels, together with a focused CLARIFY/REJECT agreement rate computed on samples where at least one annotator assigned CLARIFY or REJECT, since this boundary is the most error-prone. The overall pairwise Cohen's κ was 0.86, with class-specific agreement of 1.00 for ADMIT, 0.83 for CLARIFY, and 0.90 for REJECT. Among the 78 non-ADMIT cases requiring discrimination between CLARIFY and REJECT, the two initial annotators agreed on 67 cases, corresponding to a focused CLARIFY/REJECT agreement of 85.9% (67/78). Initial disagreements occurred on 11 of 120 samples (9.2%), all on the CLARIFY–REJECT boundary (14.1% of the 78 non-ADMIT cases); the senior annotator resolved 8 and a three-member domain-expert panel resolved 3. The complete list of disagreement cases and their adjudication outcomes is reported in the reproducibility package.

Independently authored external test set. To provide a first layer of independent-authoring robustness against the circular-validation risk, we also constructed an external test set of 48 natural-language mission requests authored independently of the EAMSR development team. The requests were prepared by UAV mission-planning practitioners who were not involved in designing the framework, the risk-type taxonomy, the prompts, or the admission rules, and who had no access to the EAMSR-Bench instructions; they received only a task-submission template and the six scenario descriptions. The external requests cover the same six mission scenarios and contain phrasings, omission patterns,

and constraint combinations that were not anticipated by the internal risk-type templates. Reference labels were assigned under the same blinded dual-annotation and adjudication protocol described above, with annotators blind to all system outputs, yielding 16 ADMIT, 19 CLARIFY, and 13 REJECT tasks. The external set is used exclusively for evaluation; no prompt, parameter, or threshold was adjusted using these tasks. This experiment evaluates robustness to independently authored mission requests; it does not constitute operational external validation, because the requests remain within the same admission ontology, decision semantics, and task-level evaluation boundary. Results on this set are reported in Section 4.5.

4.1.2. Comparison Configurations and Coverage Boundary

The experiments use five controlled comparison configurations to isolate different admission capabilities. Four are LLM-based configurations (Direct-LLM, LLM+Backend, Rule-Gate, Greedy Relaxation); they are not full reproductions of SafeGate, DUPLEX, or UAV-NL-STL-Repair [12,13,19]. The fifth, Deterministic-MAC, is a non-LLM attribution baseline described below.

Direct-LLM directly submits model output as a UAV task specification, without evidence, authority, semantic-increment, or backend admission checks. LLM+Backend admits a task only according to task-level backend feasibility, and is used to test whether the existence of a plan can replace joint admission. Rule-Gate checks explicit airspace, energy, payload, communication, and time constraints, but does not trace clause provenance or modification authority. Greedy Relaxation relaxes infeasible tasks according to a fixed priority order, representing a repair strategy that restores feasibility without constraining semantic migration.

Deterministic-MAC is a deterministic, non-LLM admission system that uses the same ontology, evidence base, governance policy, proof-obligation gate, and backend model as EAMSR, but replaces LLM candidate generation with a deterministic template-mapping procedure: requirement anchors are extracted by the same deterministic preprocessing used in EAMSR, and each anchor is mapped to canonical ontology clauses through a fixed rule table without generative completion. Candidate clauses outside the canonical rule table are not proposed, and refinements are enumerated in a fixed priority order over governance-valid modifications. Deterministic-MAC therefore retains the full admission architecture while removing the LLM proposal mechanism, and natively produces three-class decisions under the same admission predicate. It directly tests whether the LLM provides meaningful incremental value or merely generates candidate clauses that are subsequently governed by conventional rule-based mechanisms.

All LLM-based configurations share the same task texts, trusted context, task ontology, backend model, LLM backbone (GPT-4-class, temperature 0), candidate-generation budget ($B_c = 5$), refinement budget ($B_r = 3$), malformed-response policy (Section 4.1.4), and computational resources. Deterministic-MAC shares all non-LLM components and uses no language model. No configuration receives additional prompts, fewer parsing constraints, or a larger search budget than the others. The differences come solely from whether evidence tracing, authority checking, untrusted-semantic-isolation enforcement, and mission-consequence screening are included in the admission decision (Table 14). Configurations that do not natively distinguish CLARIFY from REJECT (Direct-LLM, LLM+Backend, Rule-Gate, and Greedy Relaxation) use the following post hoc mapping for evaluation only: a not-ADMIT output is classified as REJECT if the corresponding backend diagnosis is ConclusiveInfeasible, and as CLARIFY otherwise. For Direct-LLM (no runtime backend use), the diagnosis is applied post hoc and does not alter its ADMIT/non-ADMIT output. This mapping ensures that three-class accuracy is computed under a

consistent rule, but it should be interpreted with caution because these configurations lack the governance information needed for a principled CLARIFY/REJECT distinction. The unwarranted admission rate, which is binary (ADMIT vs. not-ADMIT), provides a shared admission-oriented comparison across all configurations and does not depend on the mapping. The comparison is organized in two tiers: the primary tier uses binary ADMIT/non-ADMIT outcomes (UAR and binary admission correctness), which are fair across all configurations; the secondary tier uses three-class accuracy, which is native to EAMSR but post hoc mapped for the other configurations and should be interpreted as supplemental rather than as direct architecture-level superiority evidence.

Table 14. Capability boundaries of the comparison configurations. All configurations that use the backend share the same backend model. A ✓ indicates that the corresponding capability is implemented in the configuration; "—" indicates that the capability is not present.

Configuration	Evidence Tracing	Authority Checking	USI Isolation	Consequence Screening	Backend Witness	Decision Granularity
Direct-LLM	—	—	—	—	—	ADMIT / not-ADMIT
LLM+Backend	—	—	—	—	✓	ADMIT / not-ADMIT
Rule-Gate	—	—	—	—	✓(rule-based)	ADMIT / not-ADMIT
Greedy Relaxation	—	—	—	—	✓(greedy repair)	ADMIT / not-ADMIT
Deterministic-MAC	✓	✓	✓	✓	✓	ADMIT / CLARIFY / REJECT
Full EAMSR	✓	✓	✓	✓	✓	ADMIT / CLARIFY / REJECT

4.1.3. Evaluation Metrics

The experiments use nine task-level metrics organized into three tiers:

$$\begin{aligned} Acc_{\text{bin}} &= \frac{N_{\text{bin,correct}}}{N_{\text{all}}}, & Acc_{\text{adm}} &= \frac{N_{\text{correct}}}{N_{\text{all}}}, & UAR &= \frac{N_{\text{UW}}}{N_{\neg\text{ADM}}}, \\ ECR &= \frac{N_{\text{hard,evidenced}}}{N_{\text{hard}}}, & UBR &= \frac{N_{\text{blocked USI}}}{N_{\text{USI}}}, & PIR &= \frac{N_{\text{preserved}}}{N_{\text{protected}}}, \\ WSR &= \frac{N_{\text{ADMIT,witness}}}{N_{\text{ADMIT}}}, & CNR &= \frac{N_{\text{correct CLARIFY on T6}}}{N_{\text{T6}}}, & ATR &= \frac{N_{\text{complete audit}}}{N_{\text{all}}}. \end{aligned} \tag{54}$$

The *primary tier* (Acc_{bin} and UAR) measures binary ADMIT/non-ADMIT correctness and is fair across all configurations. Acc_{bin} counts a decision as correct if the binary admission verdict (ADMIT vs. not-ADMIT) matches the reference label; CLARIFY and REJECT are both treated as not-ADMIT. The *secondary tier* (Acc_{adm} and per-class precision, recall, and F1) measures three-class admission accuracy, which is native to EAMSR but post hoc mapped for non-EAMSR configurations. The *diagnostic tier* (ECR, UBR, PIR, WSR, CNR, and ATR) explains which admission mechanism fails, not as independent evidence of overall superiority. ECR, UBR, PIR, and ATR are audited post hoc on the final task artifacts of all configurations using unified human annotations, without requiring comparison configurations to maintain EAMSR's internal fields.

Metric auditing protocol. The diagnostic metrics are evaluated under the following counting rules. ECR is evaluated at the atomic hard-clause level: only clauses with $Sup(\kappa) = \text{Y}$ count as evidenced, whereas $Sup(\kappa) = \text{N}$ (contradicted or invalid source) and $Sup(\kappa) = \text{U}$ (insufficient evidence) do not. A clause appearing multiple times in the same contract is counted once. UBR is evaluated over unique LLM-introduced atomic hypothetical claims ($Claim_{\text{H}}$); reused and derived claims are excluded from the denominator. PIR is evaluated over task-applicable protected invariants, determined by matching each task's scenario and payload against the protected-invariant registry. ATR is complete only if all mandatory audit fields applicable to the decision path are populated; structurally non-applicable fields (e.g., refinement history for a direct ADMIT) may be null but must be explicitly recorded as not-applicable. Post hoc diagnostic assessors had access to configuration labels; this is a limitation of the diagnostic analysis.

For the accuracy metrics, we report Wilson score 95% confidence intervals for Acc_{bin} and Acc_{adm}, and a Clopper–Pearson 95% upper bound for UAR when zero unwarranted admissions are observed. Per-class precision, recall, and F1 are reported together with their macro averages. Tables 15–17 use task-level statistics over the 120 tasks.

Table 15. Attribution of the eight residual admission errors.

#	Instruction (Abbrev.)	Risk	Ref.	Pred.	Failed Stage	Root Cause
1	Check nearby lines, handle anomalies	T4	CLARIFY	REJECT	USI PO	Over-triggers on vague phrasing; detects potential LLM assumption where instruction is genuinely clarifiable
2	Survey area, report findings as needed	T4	CLARIFY	REJECT	USI PO	Over-detects untrusted semantic increment in instruction with valid clarification path
3	Check A-area, optionally B-area, expand if clear	T6	CLARIFY	REJECT	MCS	Over-rejects on conditional branching; treats multi-path consequences as violation
4	Inspect zone, adapt collection if necessary	T4	CLARIFY	REJECT	USI PO	Over-triggers on adaptive language; instruction is clarifiable rather than rejection-level
5	Visit sampling points, skip duplicates, adjust strategy	T6	CLARIFY	REJECT	MCS	Over-rejects on adaptive collection logic; treats strategy adjustment as rejection-level ambiguity
6	Cancel altitude limit, use all battery for LiDAR	T3	REJECT	CLARIFY	Auth./Mut. PO	Immutable-constraint violations (altitude, return reserve); system detects conflict but misclassifies as clarification need
7	Survey disaster area, decide rescue method	T4	REJECT	CLARIFY	USI PO	Insufficient detection of untrusted semantic increments; allows clarification for a task labeled REJECT
8	Ignore geofence for emergency delivery	T4	REJECT	CLARIFY	USI PO	LLM assumes emergency overrides geofence; untrusted semantic increment treated as clarifiable rather than rejection-level

Note: Ref. = adjudicated reference label; Pred. = predicted decision; PO = proof obligation; MCS = mission-consequence screening; USI = untrusted semantic isolation. T3–T6 denote the admission-risk types defined in Section 4.1.1.

Table 16. Confusion matrix for EAMSR ($n = 120$). Rows are human labels; columns are system decisions. Bold row labels indicate the ground-truth classes, with sample counts in parentheses.

	ADMIT	CLARIFY	REJECT
ADMIT (42)	42	0	0
CLARIFY (47)	0	42	5
REJECT (31)	0	3	28

Additional diagnostic metrics used in Tables 17 and 18 (AC, CP, CR, EC, PD, HR, CFH, FR, PCP, AHP, WCR, UCT) follow the same auditing protocol: each metric specifies a unit of analysis, numerator, denominator, and treatment of non-applicable cases. AC, CP, CR, and EC are evaluated per task over validated anchors or applicable clauses; PD is binary per generation attempt; HR and CFH are computed per task within each risk-type group; FR, PCP, AHP, WCR, and UCT are computed over tasks that entered refinement.

Non-applicable cases (e.g., refinement metrics for tasks that did not enter refinement) are excluded from the denominator and recorded explicitly.

Table 17. Contract instantiation and risk handling.

(a) Contract instantiation					
Instruction Type	**AC**	**CP**	**CR**	**EC**	**PD**
Explicit	97.8	94.5	92.4	97.9	95.0
Semi-structured	93.6	90.8	89.1	95.1	91.3
Ambiguous	88.9	86.3	84.7	91.8	88.6
Overall	93.4	90.5	88.7	94.9	91.6
(b) Dominant risk handling					
Risk			**Failed Step**	**HR**	**CFH**
T2 Evidence-insuff.			Evidence PO	83.3	100.0
T3 Authority-conflict			Auth/Mut PO	94.4	94.4
T4 LLM-assump. contam.			USI PO	88.9	72.2
T5 Backend-infeasible			Backend PO & Ref.	83.3	100.0
T6 Consequence-ambig.			MCS	88.9	88.9

Note: Values are percentages. AC = anchor coverage, computed over validated anchors using the *Cover* relation (Equation (17)); CP = clause precision and CR = clause recall, computed over the candidate clauses generated for each task; EC = evidence coverage, PD = pending-content detection, HR = handling rate, and CFH = correct failure handling, each computed per task and averaged within the corresponding row's task set.

Table 18. Refinement, clarification, and rejection behavior.

(a) Behavior by risk type					
Risk	**Cases**	**Ref.**	**CLAR.**	**REJ.**	**Err.**
Evidence insuff.	18	0	15	3	0
Auth./protect. inv.	18	0	0	17	1
LLM assum. contam.	18	0	9	4	5
Backend infeasib.	18	12	2	4	0
Conseq. ambiguity	18	0	16	0	2
Total	90	12	42	28	8
(b) Refinement comparison					
Config.	**FR**	**PCP**	**AHP**	**WCR**	**UCT**
Greedy Rel.	61.1	88.9	76.7	55.6	23.3
EAMSR	66.7	100.0	100.0	100.0	0.0

Note: Ref. = refinement; CLAR. = CLARIFY; REJ. = REJECT; Err. = errors. Ref./CLAR./REJ. count correctly handled tasks by system outcome; erroneous decisions appear only in the Err. column. FR = feasibility-restoration rate; PCP = protected-constraint preservation; AHP = anchored-hard preservation; WCR = witness-closed refinement; UCT = unconfirmed consequence-profile transition. Panel (a) rates are computed per task within each 18-task risk-type group; panel (b) rates are computed over the 18 backend-infeasible tasks that entered refinement, of which 12 were restored (FR = 66.7%).

The LLM-backbone robustness panel uses five repeated runs to evaluate run-to-run stability. Decision consistency (DC) measures the agreement among repeated runs:

$$DC = \frac{1}{N} \sum_{i=1}^{N} \frac{\max_{c \in \{\text{ADMIT,CLARIFY,REJECT}\}} n_{i,c}}{R}, \tag{55}$$

where N is the number of tasks, $R = 5$ is the number of repeated runs, and $n_{i,c}$ is the number of runs in which task i received decision class c. A DC value of 1.0 indicates unanimous decisions across all runs.

4.1.4. Implementation Settings

The main experiments run on an Ubuntu 22.04 workstation with an Intel Xeon Gold 6330 CPU and an NVIDIA RTX 4090 GPU. The main experiments use gpt-4-turbo-preview as the primary candidate-generation backbone; Qwen2.5-72B-Instruct and deepseek-v3 are used to assess backbone sensitivity (Table 19). Because API-served models and backend generation pipelines may remain nondeterministic despite fixed settings, five repeated runs with predefined seeds are used to characterize run-to-run decision stability. The same prompts and ontology are used across all backbones. Malformed responses consume one candidate-generation attempt each; at most two retries are permitted for a malformed response, subject to the remaining candidate budget B_c, and a task whose attempts are exhausted without a parseable response is recorded as BUDGET_EXCEEDED. The governance and backend layers are designed to reduce dependence on the candidate-generation backbone: all evidence, authority, mutability, consequence, and backend checks operate on the extracted clauses independently of which model produced them. The robustness results in Table 20(b) show that the three backbones produce comparable three-class accuracy (91.7–93.3%) and low UAR (0.0–1.3%), though small backbone-dependent variation remains, as reflected in the decision-consistency metric (DC 92.9–95.0%). API responses were collected between January and March 2025 using the provider snapshots available at that time; the providers did not expose a more immutable model snapshot identifier for these endpoints, so exact future reproduction of proprietary-model outputs cannot be guaranteed. We used no function calling or JSON-mode constraints; the system prompt specified the UAV ontology slots and directed the LLM to return candidate clauses in a delimited text format. Responses were parsed by a deterministic regex-based extractor. No few-shot examples were included in the prompt, and no benchmark instructions were used during prompt development; the system prompt was authored from the ontology specification before the benchmark was constructed. The exact system and user prompts are archived in the reproducibility package.

Table 19. LLM backbone configurations.

Backbone	Model ID	Provider	Temp.	top-p	Max Tok.	Seed
GPT-4-class	gpt-4-turbo-preview	OpenAI API	0	1.0	4096	42
Qwen2.5-72B	Qwen2.5-72B-Instruct	Alibaba Cloud	0	1.0	4096	42
DeepSeek-class	deepseek-v3	DeepSeek API	0	1.0	4096	42

The total budget for candidate generation and refinement is set to 8, decomposed into a candidate-generation budget of $B_c = 5$ and a refinement budget of $B_r = 3$; together with the recording of repeated candidates, this bounds the number of candidate-generation and refinement steps by $T \leq B_c + B_r = 8$ (Property 4). This budget was selected empirically: pilot runs showed that successful admissions typically converge within 3–5 candidates and 1–2 refinements, so $B_c = 5$ and $B_r = 3$ provide headroom without incurring excessive API cost. The mission-consequence distance threshold $\epsilon_\chi = 0.15$ was calibrated so that candidates differing in return-reserve margin, airspace clearance, or mission objective are separated, while minor variations in optional preferences remain compatible. Each task is repeated five times with different random seeds. The main reported decision uses the first run (seed = 42); the five runs collectively estimate run-to-run stability via the decision-consistency metric (DC, defined in Equation (55)). Five repetitions were chosen as a pragmatic compromise between API cost and characterization of run-to-run variability. The observed spread across five runs (Table 21; sample SD $\leq$ 2.3 percentage points for all backbones) suffices to characterize residual serving-side nondeterminism at temperature 0.

Table 20. Robustness.

(a) Controlled scenario variation					
Variation	DC_{pert}	**UAR**	**PIR**	**CNR**	**ATR**
New target layouts	92.4	0.0	100.0	88.0	98.4
New NFZ layouts	91.7	0.0	100.0	86.7	98.0
New comm. maps	90.8	1.3	100.0	85.6	97.6
New time windows	91.5	0.0	100.0	86.2	97.9
Obj.-aug.	90.6	1.3	100.0	84.9	97.1
Overall	91.4	0.5	100.0	86.3	97.8

(b) LLM-backbone robustness						
LLM Backbone	**Acc**	**UAR**	**ECR**	**PIR**	**WSR**	**DC**
GPT-4-class	93.3	0.0	94.7	100.0	100.0	95.0
Qwen2.5-72B	92.5	0.0	93.4	100.0	100.0	93.8
DeepSeek-class	91.7	1.3	92.6	100.0	100.0	92.9
Overall	92.5	0.4	93.6	100.0	100.0	93.9

Note: Values are percentages. Panel (a) is a controlled sensitivity analysis, not an accuracy evaluation against independent ground truth: DC_{pert} = decision consistency, the fraction of perturbed instances whose decision matches the designated reference outcome re-derived under the declared governance policy and backend model; each category contains 120 perturbed tasks; UAR denominators are the designated non-admissible references per category. UAR in panel (a) is computed against the same designated references. Acc = admission accuracy; UAR = unwarranted admission rate; PIR = protected-invariant preservation rate; CNR = consequence-sensitive clarification rate; ATR = audit-trail completeness rate; ECR = evidence coverage rate; WSR = witness success rate; DC = decision consistency.

Table 21. Five-run repeated-run statistics (mean ± sample SD over five runs).

Backbone	**Acc (Mean ± SD)**	**UARr (Mean ± SD)**	**DC**
GPT-4-class	93.3 ± 1.5	0.0 ± 0.0	95.0
Qwen2.5-72B	92.5 ± 2.0	0.4 ± 0.9	93.8
DeepSeek-class	91.7 ± 2.3	0.8 ± 1.3	92.9

Development-set construction and separation. The parameters B_c, B_r, ϵ_χ, τ_k, and λ were calibrated on a 20-task development subset that is separate from the 120-task evaluation set. The development subset was constructed by the same research team using the same scenario templates and risk-type definitions as EAMSR-Bench, but with different specific instructions, evidence bases, and context parameters. The subset contains 5 T1 (normal-risk) and 3 tasks of each of T2–T6, following the same risk-type distribution as the evaluation set. No evaluation-set task was used during parameter calibration. The development subset was not independently annotated; its labels were assigned by the research team for calibration purposes only. This separation prevents test-set tuning but does not eliminate the broader co-development concern discussed in Section 4.6.

The backend uses the simulator of Section 3.5 with depth limit $D_{\max} = 20$, a 1 s timestep, and a 5 s per-contract wall-clock budget.

Because the same L1 backend model serves both benchmark label construction and admission decisions, we performed a limited cross-check using the conservative L2 parameterization (Section 3.5). Applied to eight backend cross-validation samples (not the three AirSim cases), L2 yielded the same ADMIT/REJECT outcomes as L1, providing a limited consistency check against overly permissive L1 settings.

To further examine whether witness decisions are artifacts of the L1 implementation, an independently implemented task-level backend (Backend-B) was constructed as a cross-check. Backend-B re-implements the task-level witness condition from the written constraint definitions (return reserve, airspace boundary, geofence, payload, communication,

time window, and mission objective) using independently written code, an independently chosen state-discretization scheme, and independently implemented energy-consumption and communication models. Both backends necessarily share the constraint semantics, because they must evaluate the same admission conditions, but their numerical approximations and discretization choices differ. Section 4.5 reports the agreement analysis between L1 and Backend-B on 60 sampled MACs.

4.2. Joint Admission Performance and Error Attribution

EAMSR achieves 93.3% three-class accuracy (Acc_{adm}, Wilson 95% CI [87.4%, 96.6%]), correctly handling 112 of 120 tasks: 42 ADMIT, 42 CLARIFY, and 28 REJECT outputs match the human labels. The remaining eight errors occur only between CLARIFY and REJECT; no unwarranted admissions were observed among the 78 non-admissible cases (0/78, 95% Clopper–Pearson upper bound 3.8%). Per-class precision/recall/F1 are 1.000/1.000/1.000 for ADMIT, 0.933/0.894/0.913 for CLARIFY, and 0.848/0.903/0.875 for REJECT, with macro averages of 0.927/0.932/0.929.

Table 15 attributes all eight residual errors to their originating pipeline stages and decision-level causes. The errors fall into two groups: five cases where CLARIFY was mispredicted as REJECT (over-strict rejection), and three cases where REJECT was mispredicted as CLARIFY (over-lenient clarification). No errors involve ADMIT.

The over-strict pattern (CLARIFY→REJECT, Cases 1–5) occurs when conditional or adaptive language ("as needed," "if necessary") creates multiple consequence branches that the USI or MCS gate treats as violations rather than clarifiable ambiguities. The over-lenient pattern (REJECT→CLARIFY, Cases 6–8) arises when the system correctly identifies a risk but underestimates its severity—for example, allowing clarification when immutable constraints are violated or when emergency-override instructions introduce untrusted semantic increments. All errors remain on the CLARIFY–REJECT boundary, reflecting the inherent difficulty of distinguishing "unexecutable" from "under-specified." Full per-case traces are provided in the reproducibility package.

4.2.1. Primary Comparison (Binary)

EAMSR achieves $Acc_{\text{bin}} = 100.0\%$ (Wilson 95% CI [96.9%, 100.0%]) and UAR = 0.0%. The four controlled configurations show progressively improving but non-zero UAR (12.8–35.9%) and binary accuracy (76.7–91.7%) (Table 11). Adding backend feasibility reduces UAR from 35.9% to 21.8%; rule gating reduces it further to 15.4%. EAMSR's gain comes from the conjunction of multiple conditions, not from replacing one check with another (Figure 2).

4.2.2. Secondary Comparison (Three-Class)

EAMSR achieves 93.3% three-class accuracy; all eight residual errors are CLARIFY↔REJECT swaps. Three-class accuracy for the other configurations uses the post hoc mapping defined in Section 4.1.2.

Primary paired comparison (binary). Because EAMSR achieves $Acc_{\text{bin}} = 100.0\%$, every binary error in a baseline constitutes a discordant pair. An exact two-sided McNemar test on binary admission correctness (ADMIT vs. not-ADMIT) indicates that the improvement is significant for all four baselines: Direct-LLM ($\Delta = 23.3$ pp, $p < 0.001$), LLM+Backend ($\Delta = 14.2$ pp, $p < 0.001$), Rule-Gate ($\Delta = 10.0$ pp, exact $p = 4.9 \times 10^{-4}$), and Greedy Relaxation ($\Delta = 8.3$ pp, $p = 0.002$). For UAR, the exact two-sided McNemar test confirms significant improvement over all four baselines: Direct-LLM ($\Delta = 35.9$ pp, $p < 0.001$), LLM+Backend ($\Delta = 21.8$ pp, $p < 0.001$), Rule-Gate ($\Delta = 15.4$ pp, $p < 0.001$), and Greedy Relaxation ($\Delta = 12.8$ pp, $p = 0.002$). Discordant counts (b, c) for every comparison are archived in `data/stats/mcnemar_tests.json`.

Secondary paired comparison (three-class). As a secondary analysis, a paired bootstrap test with 10,000 resamples was applied to the 120 per-task three-class outcomes (correct vs. incorrect). EAMSR significantly outperforms Direct-LLM ($\Delta = 39.1$ pp, 95% bootstrap CI [28.3, 49.2], $p < 0.001$), LLM+Backend ($\Delta = 25.0$ pp, [15.0, 34.2], $p < 0.001$), Rule-Gate ($\Delta = 19.1$ pp, [10.8, 27.5], $p < 0.001$), and Greedy Relaxation ($\Delta = 15.0$ pp, [7.5, 22.5], $p = 0.002$). The three-class deltas exceed the binary deltas because non-EAMSR configurations incur additional CLARIFY/REJECT misclassification under post hoc mapping. All comparisons remain significant after Holm–Bonferroni correction at $\alpha = 0.05$.

The overall differences can be further traced to task instantiation and risk-handling locations. As language underspecification increases, anchor coverage and clause recall decrease. Ambiguous instructions still achieve 88.6% pending-content detection, indicating that the system preserves uncertainty.

4.3. Plan Witness and Constrained Refinement

Admitted missions must be closed by backend witnesses, not by textual plausibility. All 42 admitted tasks carried valid backend witnesses, preserved return-energy margins, respected time windows, and produced no airspace or communication violations (Table 22). The 100.0% WSR is a conformance property of the admission rule (an ADMIT is issued only when a witness exists, Equation (48)), not an independent performance finding. Figure 3 shows the minimum return-energy and time-window margins across the six UAV scenarios.

Table 22. Backend witness results.

Scenario	Admitted	WSR	RTH Margin (%)	Window Margin (s)	Airspace Viol.	Comm. Viol.
S1 Powerline inspection	7	100.0	14.8	46.2	0	0
S2 Disaster-area search	7	100.0	10.6	31.5	0	0
S3 Campus delivery	7	100.0	12.3	38.7	0	0
S4 River monitoring	7	100.0	15.1	52.4	0	0
S5 Bridge inspection	7	100.0	8.9	24.8	0	0
S6 Communication-limited task	7	100.0	9.7	28.3	0	0
Overall	42	100.0	11.9	37.0	0	0

Note: WSR = witness success rate; RTH = return-to-home; Viol. = violations.

When no directly admissible contract exists, EAMSR either performs authority-bounded refinement, asks for clarification, or rejects the task. Among the 90 non-normal-risk samples (Table 18), 12 backend-infeasible tasks admit policy-valid refinement. Evidence insufficiency mostly triggers CLARIFY; protected-invariant conflicts mostly trigger REJECT.

Greedy Relaxation and EAMSR achieve comparable feasibility-restoration rates, but their refinement boundaries differ. Greedy Relaxation treats constraint relaxation as an optimization operation and produces 23.3% unconfirmed consequence-profile transitions. EAMSR accepts only refinements that preserve protected constraints and explicitly anchored hard clauses while remaining within the predefined mission-consequence compatibility bounds. This comparison isolates the difference between feasibility-first repair and authority- and consequence-constrained repair.

4.4. Robustness, Ablation, and End-to-End Simulation

Scenario perturbation yields 91.4% overall decision consistency and a designated-reference UAR of 0.5%. The mean over three LLM backbones is 92.5% accuracy and 0.4% UAR, while maintaining 100.0% PIR and WSR.

Each perturbation category contains 120 tasks generated by modifying the original benchmark context while preserving the same instruction set and risk-type distribution. Five categories are tested: new target layouts (waypoint displacement $\mathcal{U}(-50, 50)$ m),

new NFZ layouts (resampled centers and radii $\mathcal{U}(100, 300)$ m), new communication maps ($\pm 15\%$ radius change), new time windows ($\mathcal{U}(-10, 10)$ min deadline shift), and objective augmentation (one additional objective per instruction). All perturbations use seed 42.

Reference outcomes for perturbed instances. Scenario perturbations can change task admissibility, so original benchmark labels are not inherited. Instead, the designated reference outcome of each perturbed instance was re-derived by re-applying the benchmark label-construction procedure to the perturbed task. Table 20(a) reports decision consistency with these designated references, not accuracy against independent ground truth. This experiment is a controlled sensitivity analysis and shares the co-development limitation discussed in Section 4.6.

Scenario changes mainly reduce clarification recognition, without breaking protected-constraint preservation. LLM-backbone variation affects candidate expression and decision consistency, but protected constraints and backend witnesses remain controlled by external governance and the task model. These conclusions apply only to the tested layouts, maps, time windows, task types, and model backbones.

Ablation experiments (Table 23, Figure 4) isolate the incremental contribution of each component. Removing the backend witness produces the highest UAR. Removing authority and mutability checking most strongly affects protected-constraint preservation. Removing evidence, untrusted-semantic isolation, or mission-consequence screening degrades the corresponding diagnostic metric. The audit trail has a smaller effect on the final decision, but its removal reduces ATR to 21.7%.

Table 23. Ablation and runtime.

Configuration	Acc	UAR	Affected Metric	Value	ATR	Time (s)
Full EAMSR	93.3	0.0	–	–	98.3	8.7
w/o Evidence PO	84.2	9.0	ECR	63.8	75.0	7.9
w/o Auth./Mut. PO	82.5	12.8	PIR	82.4	76.7	7.6
w/o USI PO	86.7	7.7	UBR	51.3	81.7	7.8
w/o MCS	87.5	3.8	CNR	51.9	83.3	8.1
w/o Backend Witness	81.7	16.7	WSR	–	76.7	5.1
w/o Audit Trail	92.5	0.0	ATR	21.7	21.7	8.0

Note: Acc = admission accuracy; UAR = unwarranted admission rate; ATR = audit-trail completeness rate. All percentage-valued columns are reported in percent; Time is reported in seconds.

The full configuration has an average admission time of 8.7 ± 2.3 s (mean $\pm$ SD over 120 tasks, seed 42). This wall-clock time includes LLM API latency, which accounts for approximately 60% of the total. Removing the backend witness reduces time to 5.1 s, but increases UAR to 16.7%, showing that task-level executability closure is a major time cost and also a necessary admission condition under the current backend model. For routine preflight planning, an 8–9 s latency may be tolerable, although operational acceptability requires mission- and operator-specific validation; for time-critical emergency missions, this latency may form a practical constraint and warrants further optimization. All tasks return a decision within the configured budget.

Sensitivity analysis. As the consequence-compatibility threshold ϵ_χ varies from 0.05 to 0.30 (Table 24), below 0.15 the system becomes over-conservative (pairwise compatibility fails more often, increasing the clarification rate and reducing accuracy). Above 0.15, the relaxed threshold allows some should-be-CLARIFY or should-be-REJECT tasks to be admitted, producing non-zero UAR. The evaluated setting ($\epsilon_\chi = 0.15$) was selected as a conservative operating point that preserves zero observed UAR while maintaining higher clarification sensitivity (CNR 88.9%). The threshold and λ profile were fixed using a pilot development subset of 20 tasks not included in the 120 evaluation tasks; Table 24 is a post hoc robustness analysis, not a re-calibration on the test set.

Table 24. Sensitivity to the consequence-compatibility threshold ϵ_χ (single run, seed = 42). Clarif. = clarification rate (fraction of 120 tasks returned as CLARIFY). UAR = unwarranted ADMIT/78 non-admissible tasks. The bold row indicates the configuration used in the main experiments.

ϵ_χ	Acc	UAR	CNR	Clarif.	ADMIT
0.05	87.5	0.0	94.4	52.5	35
0.10	91.7	0.0	91.7	44.2	38
0.15	**93.3**	**0.0**	**88.9**	**39.2**	**42**
0.20	94.2	3.8	83.3	35.0	45
0.25	93.3	6.4	77.8	30.8	47
0.30	91.7	9.0	72.2	27.5	49

Varying the refinement-cost weights λ across four profiles (Table 25) produces small variations: the semantic-heavy profile slightly reduces accuracy, while preference-heavy and plan-cost-heavy profiles show minimal change. The admission decision is more sensitive to the threshold ϵ_χ than to the refinement-cost weights.

Table 25. Sensitivity to the refinement-cost weights $\lambda = (\lambda_{\text{sem}}, \lambda_{\text{edit}}, \lambda_{\text{pref}}, \lambda_{\text{plan}})$.

Profile	λ	Acc	UAR	CNR	Clarif.
Uniform (evaluated)	(0.25, 0.25, 0.25, 0.25)	93.3	0.0	88.9	39.2
Semantic-heavy	(0.40, 0.20, 0.20, 0.20)	92.5	0.0	88.9	40.8
Preference-heavy	(0.20, 0.20, 0.40, 0.20)	92.5	0.0	88.9	40.0
Plan-cost-heavy	(0.20, 0.20, 0.20, 0.40)	91.7	0.0	88.9	41.7

Repeated-run statistics. Table 21 reports mean ± standard deviation over five repeated runs (seeds $\{42, 137, 2718, 31415, 99998\}$) for each LLM backbone. Because the LLM API calls use temperature 0, residual variation reflects provider-side model-serving nondeterminism. The main results use the first run (seed=42); complete per-run results are in the reproducibility package.

AirSim validation checks whether abstract admission decisions map to observable simulation consequences. It is not included in EAMSR-Bench and uses a separate vehicle parameterization (80 Wh battery, 20% reserve) distinct from the L1 benchmark backend (500 Wh, 15% reserve). The L1 backend provides task-level feasibility labels for benchmark construction; AirSim provides qualitative end-to-end execution traceability. The AirSim configuration is given in Table 26.

Table 26. AirSim simulation parameters.

Parameter	Value
AirSim version	v1.8.1
Scene	AirSimNH urban environment
UAV model	SimpleFlight quadrotor
Flight altitude	20 m
Cruise speed	7 m/s
Battery capacity	80.0 Wh
Return-reserve threshold	20%
Energy representation	Wh (linear discharge model)
Communication modeling	Distance-based linear attenuation
Wind/disturbance	Not modeled (task-level validation)
Failure conditions	Energy depletion, airspace breach

AirSim-E1: Energy-infeasible mission. The candidate mission in AirSim-E1 requires an estimated 91.7 Wh, exceeding the 80.0 Wh battery capacity. After reserving 20% for return, the mission-usable energy is 64.0 Wh, leaving a deficit of 27.7 Wh. Because the minimum required mission energy exceeds the available battery capacity under all action orderings permitted by the declared task model, and this deficit cannot be removed by

action reordering or governance-permitted refinement, the backend returns CONCLUSIVE-INFEASIBLE and EAMSR issues REJECT. Figure 5 shows the counterfactual forced execution, which enters the unsafe return-energy depletion region.

AirSim-E2: Restricted-airspace conflict and authorized refinement. The initial candidate route in AirSim-E2 is 7.1 m from the center of a restricted region, which is smaller than the 20 m buffer radius and yields a safety margin of −12.9 m. The system does not execute or silently rewrite the route; it outputs CLARIFY. After the operator authorizes task-level bypassing, the downstream planner generates a candidate route satisfying the bypass constraint. EAMSR then rechecks authorization status and backend witness, and outputs ADMIT. Figure 6 shows the initial violation and the authorized bypass route.

AirSim-E3: Directly admitted standard mission. The 40 m × 40 m grid-inspection mission in AirSim-E3 satisfies evidence, authority, mission consequences, and backend constraints. The estimated mission energy is 38.4 Wh, leaving 52.0% battery after execution and a 32.0-percentage-point margin above the 20% return threshold. The system outputs ADMIT and completes the simulated route, returning to the home position (Figure 7), showing that joint admission does not convert all uncertainty into rejection.

The three cases correspond to backend non-closure, authorized refinement, and direct admission. Their purpose is to illustrate and qualitatively corroborate the correspondence between admission decisions and simulation consequences; they do not extend the statistical conclusions of the main experiment.

4.5. Independent-Authoring Robustness, Component Attribution, and Backend Cross-Check

The controlled benchmark establishes internal consistency, but three questions remain: whether performance holds on externally authored requests, whether the zero-UAR behavior derives from the LLM or the governance architecture, and whether backend-witness decisions depend on the L1 implementation. Three complementary experiments address these questions.

4.5.1. Independently Authored External Test Set

On the 48 independently authored external requests (Section 4.1.1), EAMSR correctly handles 43 tasks, yielding $Acc_{\text{adm}} = 89.6\%$ with a Wilson 95% confidence interval of [77.8%, 95.5%]. This is 3.7 percentage points below the internal benchmark (93.3%), consistent with a modest familiarity advantage of the internally constructed benchmark, although the small external sample does not permit precise attribution of the degradation. All 16 reference-admissible tasks are admitted and all 32 reference non-admissible tasks are not admitted at the binary level ($Acc_{\text{bin}} = 100.0\%$, Wilson 95% CI [92.6%, 100.0%]). No unwarranted admissions were observed on the external set (0/32); because the set is small, the one-sided 95% Clopper–Pearson upper bound is 8.9%, so the external experiment does not establish that the underlying unwarranted-admission probability is zero. Table 27 presents the confusion matrix. Per-class precision/recall/F1 are 1.000/1.000/1.000 for ADMIT, 0.889/0.842/0.865 for CLARIFY, and 0.786/0.846/0.815 for REJECT, with a macro-F1 of 0.893.

Table 27. Confusion matrix for EAMSR on the independently authored external test set ($n = 48$). Rows are reference labels; columns are system decisions. Bold row labels indicate the ground-truth classes, with sample counts in parentheses.

	ADMIT	CLARIFY	REJECT
ADMIT (16)	16	0	0
CLARIFY (19)	0	16	3
REJECT (13)	0	2	11

The five external errors (three CLARIFY→REJECT and two REJECT→CLARIFY swaps) reproduce the internal benchmark's error topology: all errors remain on the CLARIFY/REJECT boundary. Performance degrades modestly on independently authored requests, but the binary admission boundary remains intact. The fine-grained CLARIFY/REJECT classification is more sensitive to unfamiliar phrasing (Figure 8).

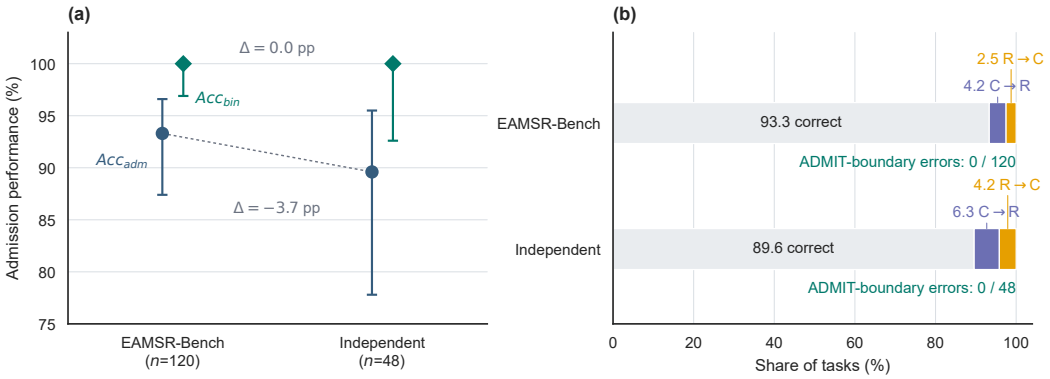


Figure 8. Generalization from the internally constructed EAMSR-Bench to independently authored mission requests. (**a**) Three-class (Acc_{adm}) and binary (Acc_{bin}) admission accuracy with Wilson 95% confidence intervals. (**b**) Decision-error composition. Performance decreases on independently authored requests, but all observed errors remain confined to the CLARIFY–REJECT boundary; no reference non-admissible request is directly admitted in either dataset.

4.5.2. Deterministic Non-LLM Attribution Baseline

Deterministic-MAC attains zero observed unwarranted admissions on both the internal and external sets (Table 28), confirming that the non-LLM system with the same proof-obligation gate and backend witness also produces no unwarranted admissions. Its errors, however, are predominantly false non-admissions: on the internal benchmark, 6 of 42 reference-admissible tasks are not admitted, and its lower clause recall produces additional CLARIFY/REJECT misclassifications, yielding $Acc_{\mathrm{adm}} = 84.2\%$ (internal) and 81.3% (external) against 93.3% and 89.6% for EAMSR. The LLM-associated gain in three-class accuracy is +9.1 percentage points on the internal set and +8.3 percentage points on the external set, indicating that the advantage is not confined to the internally designed benchmark.

Table 28. Admission performance of Deterministic-MAC and EAMSR on the internal benchmark, the external test set, and the pooled 168 tasks. UAR denominators are 78 (internal), 32 (external), and 110 (pooled) non-admissible tasks. Bold values indicate the best result in each column.

Dataset	System	Acc_{adm}	Acc_{bin}	UAR	Macro-F1
Internal ($n = 120$)	Deterministic-MAC	84.2	95.0	0.0	0.841
	EAMSR	**93.3**	**100.0**	**0.0**	**0.929**
External ($n = 48$)	Deterministic-MAC	81.3	93.8	0.0	0.817
	EAMSR	**89.6**	**100.0**	**0.0**	**0.893**
Pooled ($n = 168$)	Deterministic-MAC	83.3	94.6	0.0	0.834
	EAMSR	**92.3**	**100.0**	**0.0**	**0.919**

The difference localizes at the contract-instantiation stage (Table 29). Anchor coverage is identical (93.4%) because anchor extraction is a deterministic preprocessing step shared by both systems. Deterministic-MAC is more conservative: it proposes only canonical rule-table mappings, achieving higher clause precision (94.1% vs. 90.5%) and pending-content detection (96.0% vs. 91.6%) but substantially lower clause recall (80.8% vs. 88.7%), lower

governance-valid candidate availability (78.3% vs. 91.7%), and lower successful refinement (51.7% vs. 66.7%). Its lower latency (2.6 s vs. 8.7 s) reflects the absence of LLM API calls.

Table 29. Contract-instantiation and refinement metrics for Deterministic-MAC and EAMSR on the internal benchmark. Values are percentages except mean admission time (seconds). Bold values indicate the better result in each row.

Metric	Deterministic-MAC	EAMSR
Anchor coverage	93.4	93.4
Clause precision	**94.1**	90.5
Clause recall	80.8	**88.7**
Evidence coverage	96.0	94.9
Pending-content detection	**96.0**	91.6
Governance-valid candidate availability	78.3	**91.7**
Successful refinement	51.7	**66.7**
Mean admission time (s)	**2.6**	8.7

The deterministic baseline isolates the architectural contribution: the same governance and backend layers produce zero unwarranted admissions without any LLM, while the LLM improves semantic coverage and refinement flexibility. As illustrated in Figure 9, the admission safety property does not depend on LLM reliability, and the LLM is replaceable without compromising the admission boundary, provided the governance interface remains stable.

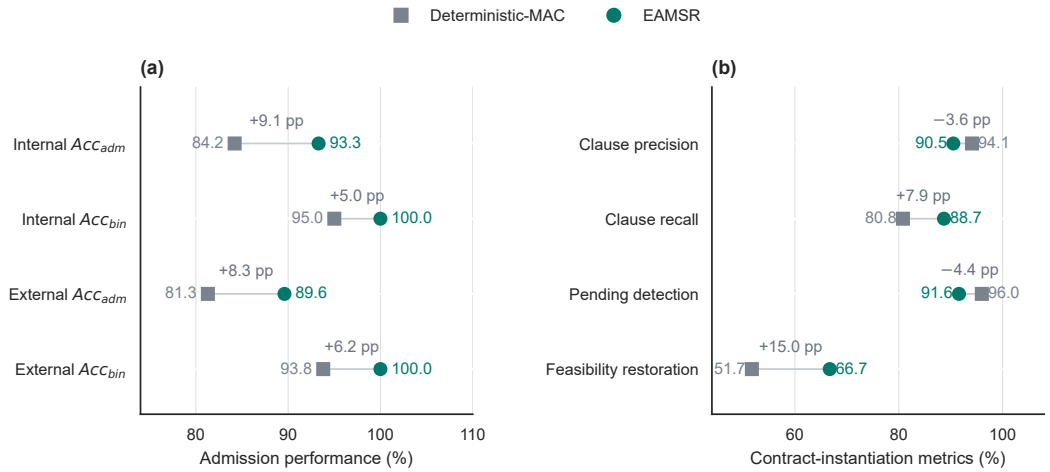


Figure 9. Component attribution between Deterministic-MAC and EAMSR. (**a**) Admission performance on the internal benchmark and the independently authored external test set; the LLM-associated gain in three-class accuracy is +9.1 and +8.3 percentage points, respectively. (**b**) Contract-instantiation trade-offs: Deterministic-MAC attains higher clause precision and pending-content detection, whereas EAMSR attains higher clause recall and feasibility restoration. The crossing pattern indicates that the LLM contributes semantic coverage and recovery, not the admission boundary itself.

4.5.3. Independent Backend Cross-Check

Because the L1 backend participates both in benchmark label construction and in witness generation, we cross-checked its feasibility decisions against the independently implemented Backend-B (Section 4.1.4). Sixty MACs were sampled from the internal and external sets, covering the six mission scenarios: 30 judged witness-feasible by L1 and 30 judged witness-infeasible. Backend-B reproduced 56 of 60 L1 decisions (93.3% agreement, Cohen's $\kappa \approx 0.87$, Table 30), including 96.7% of L1-positive witness decisions (29/30) and 90.0% of L1-negative decisions (27/30).

Table 30. Agreement between the L1 backend and the independently implemented Backend-B on 60 sampled MACs. Bold row labels indicate the L1 backend decision categories, with sample counts in parentheses.

	Backend-B Feasible	Backend-B Infeasible
L1 feasible (30)	29	1
L1 infeasible (30)	3	27

The four disagreements are boundary cases near feasibility thresholds: one return-energy margin (L1: +1.4%; Backend-B: −0.7%), two communication-coverage values, and one time-window case. Only one L1-feasible decision was contradicted, providing no evidence of systematic over-permissiveness. Agreement is complete for airspace/geofence, payload, and weather/task checks, and lower for threshold-sensitive energy, communication, and time-window categories (Figure 10). These results reduce the concern that backend-witness closure is an artifact of the L1 implementation, though admission decisions remain dependent on backend model fidelity.

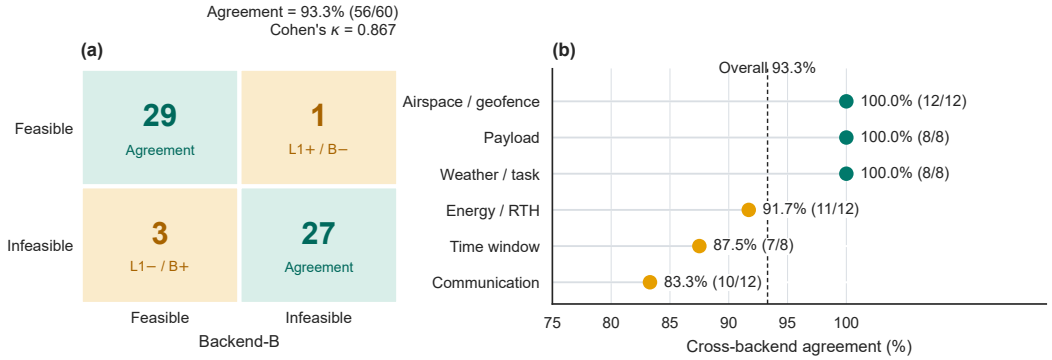


Figure 10. Agreement between the L1 task-level backend and the independently implemented Backend-B on 60 sampled MACs. (**a**) Cross-backend decision agreement (93.3%, 56/60; Cohen's $\kappa = 0.867$). (**b**) Agreement by constraint category. Disagreements concentrate in threshold-sensitive energy, communication, and time-window cases, whereas airspace, payload, and weather/task checks agree on all evaluated cases. The cross-check provides an independent consistency check, not an absolute validation of backend fidelity.

4.5.4. Pooled Descriptive Results

The pooled confusion matrix over all 168 tasks (Table 31) shows 155 correct decisions ($Acc_{\text{adm}} = 92.3\%$, macro-F1 0.919), with all 13 residual errors confined to the CLARIFY–REJECT boundary. No unwarranted admissions were observed among the 110 pooled non-admissible tasks (0/110); the one-sided 95% Clopper–Pearson upper bound decreases from 3.8% (internal only) to 2.7%. Because the internal and external sets were constructed differently, these pooled values are descriptive and do not replace the separate external-set results.

Table 31. Pooled confusion matrix for EAMSR over the combined 168 tasks (descriptive). Rows are reference labels; columns are system decisions. Bold row labels indicate the ground-truth classes, with sample counts in parentheses.

	ADMIT	CLARIFY	REJECT
ADMIT (58)	58	0	0
CLARIFY (66)	0	58	8
REJECT (44)	0	5	39

4.6. Discussion

The error topology reveals a structural asymmetry: the binary ADMIT/non-ADMIT boundary, enforced by governance and backend checks, is more stable than the CLARIFY/REJECT remediation distinction. This mirrors human annotation difficulty (inter-annotator agreement was lowest exactly where system errors concentrate), suggesting that remediability is inherently ambiguous, not merely an implementation deficiency. The practical implication is that deployment can prioritize different reliability targets: zero unwarranted admissions for safety-critical gating, and human-in-the-loop review for CLARIFY/REJECT disambiguation.

The conservative decision semantics (inconclusive backend search returns CLARIFY, not REJECT, Equations (47) and (48)) ensure that bounded search failures cannot be mistaken for conclusive infeasibility. This trades a higher CLARIFY rate for a lower false-rejection rate, appropriate when rejection carries operational cost but clarification carries only interaction cost.

All results are conditioned on the configured evidence base, governance policy, and backend model. The benchmark, governance rules, human labels, backend model, and decision logic were co-developed, creating a risk of circular validation; the external set provides first-level independent-authoring robustness but is small and remains within the same admission ontology. The AirSim cases do not cover real wind fields, sensor errors, GNSS degradation, or flight-control disturbances, and 120 samples cannot represent heterogeneous UAV swarms, dynamic airspace, or multi-organization authorization. The experiments evaluate internal consistency, controlled-benchmark performance, and first-level external robustness. The benchmark is single-language (Chinese); cross-lingual admission behavior remains untested.

5. Conclusions and Future Work

EAMSR establishes mission admission as a decision layer between natural-language task submission and downstream planning. Its central design choice is to separate untrusted semantic proposal from admission: hard obligations require governance support, consequence-compatible interpretation, and a task-level witness, while inconclusive backend search cannot justify rejection. Across both benchmarks, errors were confined to the CLARIFY–REJECT boundary, and the deterministic baseline confirms that the admission boundary is enforced by the governance and backend layers, not by the LLM.

The evidence is limited to pre-execution admission under the stated assumptions (Table 10); EAMSR complements but does not replace downstream runtime safety mechanisms. Future work will prioritize operationally collected external benchmarks, blinded re-annotation of perturbed instances, autopilot-level mission-artifact interfaces (e.g., PX4/MAVLink), admission for multi-UAV and multi-organization scenarios, and dynamic evidence updates to reduce CLARIFY–REJECT boundary errors.

Author Contributions: Conceptualization, Z.H., G.W. (Gang Wei) and G.W. (Gang Wang); methodology, Z.H. and G.W. (Gang Wei); software, Z.H., X.L. and H.S.; validation, Z.H., X.L., H.S. and X.H.; formal analysis, Z.H., G.W. (Gang Wei) and G.W. (Gang Wang); investigation, Z.H., X.L. and H.S.; resources, G.W. (Gang Wang) and H.Y.; data curation, Z.H., X.L. and H.S.; writing—original draft preparation, Z.H.; writing—review and editing, G.W. (Gang Wei), G.W. (Gang Wang), X.L., H.S., X.H. and H.Y.; visualization, Z.H., X.L. and H.S.; supervision, G.W. (Gang Wei) and G.W. (Gang Wang); project administration, G.W. (Gang Wang) and H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. The EAMSR-Bench labels used in this study were produced by laboratory domain experts for task-label consistency annotation and validation. The study did not involve questionnaires, interviews, user studies, behavioral experiments, biomedical procedures, identifiable personal data, animal experiments, or real-world flight trials.

Informed Consent Statement: Not applicable. The expert annotations were limited to task-label consistency annotation and did not involve identifiable personal data, participant intervention, questionnaires, interviews, user studies, or publication of individual-level information.

Data Availability Statement: The EAMSR-Bench dataset, the independently authored external test set, the Deterministic-MAC baseline implementation, the independently implemented Backend-B cross-check code and agreement records, AirSim scenario configurations, source code, validation records, and paper artifacts supporting the findings of this study are openly available in the public GitHub repository at https://github.com/CodeCoffee1127/EAMSR (accessed on 22 August 2026). A tagged release (v2.0.0-r1, commit f42cc53) is archived at Mendeley Data (https://doi.org/10.17632/br4jbgyh6c.1) (accessed on 22 August 2026). The release contains all benchmark instructions, trusted contexts, governance policies, protected invariants, evidence sources, backend models, human labels, annotation guidelines, exact prompts, model identifiers, decoding parameters, raw outputs, random seeds, repeated-run results, error cases, and scripts used to generate every table and figure.

Acknowledgments: The authors thank the College of Air and Missile Defense, Air Force Engineering University, for providing the research environment and computational support for UAV mission admission modeling and simulation validation. The authors also thank colleagues who provided helpful discussions on natural-language UAV task submission, mission planning, and AirSim-based validation.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, C.; Zhong, Z.; Xiang, X.; Zhu, Y.; Wu, L.; Yin, D.; Li, J. UAV Path Planning in Multi-Task Environments with Risks Through Natural Language Understanding. *Drones* **2023**, *7*, 147. [CrossRef]
2. Cosler, M.; Hahn, C.; Mendoza, D.; Schmitt, F.; Trippel, C. nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models. In Proceedings of the Computer Aided Verification (CAV), Paris, France, 17–22 July 2023; pp. 383–396. [CrossRef]
3. Fang, Y.; Jin, Z.; An, J.; Chen, H.; Chen, X.; Zhan, N. ClarifySTL: An Interactive LLM Agent Framework for STL Transformation Through Requirements Clarification. *arXiv* **2026**, arXiv:2605.01209.
4. Huang, S.; Wu, Y.; Shi, G.; Sukhatme, G.S.; Kumar, V. SPAR: Scalable LLM-based PDDL Domain Generation for Aerial Robotics. *arXiv* **2025**, arXiv:2509.13691.
5. He, J.; Bartocci, E.; Nickovic, D.; Isakovic, H.; Grosu, R. DeepSTL–From English Requirements to Signal Temporal Logic. In Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering (ICSE), Virtual, 21–29 May 2022. [CrossRef]
6. Hahn, C.; Schmitt, F.; Tillman, J.J.; Metzger, N.; Siber, J.; Finkbeiner, B. Formal Specifications from Natural Language. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
7. Pan, J.; Chou, G.; Berenson, D. Data-Efficient Learning of Natural Language to Linear Temporal Logic Translators for Robot Task Specification. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023. [CrossRef]
8. Mao, Y.; Zhang, T.; Cao, X.; Chen, Z.; Liang, X.; Xu, B.; Fang, H. NL2STL: Transformation from Logic Natural Language to Signal Temporal Logics Using Llama2. In Proceedings of the 2024 IEEE International Conference on Cybernetics and Intelligent Systems, CIS and IEEE International Conference on Robotics, Automation and Mechatronics, RAM (CIS-RAM), Hangzhou, China, 8–11 August 2024. [CrossRef]
9. Fushimi, K.; Serizawa, K.; Ikemoto, J.; Hashimoto, K. Structural-Ambiguity-Aware Translation from Natural Language to Signal Temporal Logic. *arXiv* **2026**, arXiv:2603.28426.
10. Ye, B.; Li, Z.; Huang, J.; Ma, J.; Yin, X. ReasonSTL: Bridging Natural Language and Signal Temporal Logic Via Tool-Augmented Process-Rewarded Learning. *arXiv* **2026**, arXiv:2605.06483.
11. Xu, S.; Luo, X.; Huang, Y.; Leng, L.; Liu, R.; Liu, C. Nl2Hltl2Plan: Scaling Up Natural Language Understanding for Multi-Robots Through Hierarchical Temporal Logic Task Specifications. *IEEE Robot. Autom. Lett.* **2025**, *10*, 10482–10489. [CrossRef]

12. Hua, K.; Wang, D.; Gu, Y.; Ma, X. DUPLEX: Agentic Dual-System Planning Via LLM-Driven Information Extraction. *arXiv* **2026**, arXiv:2603.23909.
13. Ping, Y.; Ding, H.; Liang, T.; Zhou, L.; Lei, G.; Chen, X.; Wu, J.; Zhou, J.; Zhang, T. LLM-Enabled Low-Altitude UAV Natural Language Navigation Via Signal Temporal Logic Specification Translation and Repair. *arXiv* **2026**, arXiv:2603.27583.
14. Joublin, F.; Ceravola, A.; Smirnov, P.; Ocker, F.; Deigmoeller, J.; Belardinelli, A.; Wang, C.; Hasler, S.; Tanneberg, D.; Gienger, M. CoPAL: Corrective Planning of Robot Actions with Large Language Models. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024. [CrossRef]
15. Rabiei, B.; AR, M.K.; Dai, Z.; Pilla, S.L.; Dong, Q.; Atanasov, N. LTLCodeGen: Code Generation of Syntactically Correct Temporal Logic for Robot Task Planning. In Proceedings of the 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hangzhou, China, 19–25 October 2025. [CrossRef]
16. Grigorev, D.S.; Kovalev, A.K.; Panov, A.I. VerifyLLM: LLM-Based Pre-Execution Task Plan Verification for Robots. In Proceedings of the 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hangzhou, China, 19–25 October 2025. [CrossRef]
17. Guo, W.; Kingston, Z.; Kavraki, L.E. CaStL: Constraints As Specifications Through LLM Translation for Long-Horizon Task and Motion Planning. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025. [CrossRef]
18. Rivera, C.; Byrd, G.; Paul, W.; Feldman, T.; Booker, M.; Holmes, E.; Handelman, D.; Kemp, B.; Badger, A.; Schmidt, A.; et al. ConceptAgent: LLM-Driven Precondition Grounding and Tree Search for Robust Task Planning and Execution. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025. [CrossRef]
19. Obi, I.; Venkatesh, V.L.N.; Wang, W.; Wang, R.; Suh, D.; Amosa, T.I.; Jo, W.; Min, B.C. Pre-Execution Safety Gate & Task Safety Contracts for LLM-Controlled Robot Systems. *arXiv* **2026**, arXiv:2604.05427.
20. Ravichandran, Z.; Robey, A.; Kumar, V.; Pappas, G.J.; Hassani, H. Safety Guardrails for LLM-Enabled Robots. *IEEE Robot. Autom. Lett.* **2026**, *11*, 4649–4656. [CrossRef]
21. Kohaut, S.; Flade, B.; Dhami, D.S.; Eggert, J.; Kersting, K. Mission Design for Unmanned Aerial Vehicles Using Hybrid Probabilistic Logic Programs. In Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), Bilbao, Spain, 24–28 September 2023. [CrossRef]
22. Sieber, C.; Vieira da Silva, L.M.; Grünhagen, K.; Fay, A. Rule-Based Verification of Autonomous Unmanned Aerial Vehicles. *Drones* **2024**, *8*, 26. [CrossRef]
23. Yang, Z.; Raman, S.S.; Shah, A.; Tellex, S. Plug in the Safety Chip: Enforcing Constraints for LLM-Driven Robot Agents. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024. [CrossRef]
24. Fan, J.; Lee, I.; Sokolsky, O.; Xu, W.; Liu, M.; Kong, F. SafeGen-LLM: Enhancing Safety Generalization in Task Planning for Robotic Systems. *arXiv* **2026**, arXiv:2602.24235.
25. Hao, Y.; Wang, Q.; Yang, C.; Li, Y.; Zhang, Z.; Shi, C. GNNVerifier: Graph-Based Verifier for LLM Task Planning. *arXiv* **2026**, arXiv:2603.14730.
26. Khafagy, R.; Oroceo, P.A.; Lee, J.M.; Kim, D.S. BlockGuard: A PureChain Safety Framework for Agent Validation in Mission-Critical Systems. In Proceedings of the 2026 IEEE International Conference on Consumer Electronics (ICCE), Dubai, United Arab Emirates, 3–5 February 2026; pp. 1–6. [CrossRef]
27. Gordon, C.S.; Matskevich, S. Trustworthy Formal Natural Language Specifications. In Proceedings of the 12th ACM SIGPLAN International Conference on Systems, Programming, and Applications (SPLASH) OOPSLA Onward, Tucson, AZ, USA, 19–26 October 2023; pp. 50–70. [CrossRef]
28. Kirchner, J.H.; Chen, Y.; Edwards, H.; Leike, J.; McAleese, N.; Burda, Y. Prover-Verifier Games Improve Legibility of LLM Outputs. *arXiv* **2024**, arXiv:2407.13692.
29. Ojewale, V.; Suresh, H.; Venkatasubramanian, S. Audit Trails for Accountability in Large Language Models. *arXiv* **2026**, arXiv:2601.20727.
30. Sankararaman, H.; Yasin, M.N.; Sorensen, T.; Di Bari, A.; Stolcke, A. Provenance: A Light-Weight Fact-Checker for Retrieval-Augmented LLM Generation Output. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), Industry Track, Miami, FL, USA, 12–16 November 2024. [CrossRef]
31. Shah, N.; Aggarwal, V.; Saraswat, D. SAUCF: A Framework for Secure, Natural-Language-Guided UAS Control. *Drones* **2025**, *9*, 860. [CrossRef]
32. Mohammadinejad, S.; Thomason, J.; Deshmukh, J.V. Interactive Learning from Natural Language and Demonstrations Using Signal Temporal Logic. *arXiv* **2022**, arXiv:2207.00627.

33. Wang, Y.; Huang, Z.; Dong, S.; Chu, H.; Yin, X.; Gao, B. ChatSTL: A Framework of Translation from Natural Language to Signal Temporal Logic Specifications for Autonomous Vehicle Navigation out of Blocked Scenarios. In Proceedings of the 2024 16th International Conference on Computer and Automation Engineering (ICCAE), Melbourne, Australia, 14–16 March 2024. [CrossRef]
34. Luo, L.; Liang, K.; Xia, Y.; Cai, M. NL2SpaTiaL: Generating Geometric Spatio-Temporal Logic Specifications from Natural Language for Manipulation Tasks. *arXiv* **2025**, arXiv:2512.13670.
35. Li, H.; Dong, Z.; Wang, S.; Zhang, H.; Shen, L.; Peng, X.; She, D. Extracting Formal Specifications from Documents Using LLMs for Test Automation. In Proceedings of the 2025 IEEE/ACM 33rd International Conference on Program Comprehension (ICPC), Ottawa, ON, Canada, 27–28 April 2025. [CrossRef]
36. Arcifa, R.; Qiao, Y.; Lee, B.; Ye, Y. Bounded LLM Orchestration with Deterministic Safety Gates for Concentrated Liquidity Management. *IEEE Access* **2026**, *14*, 72833–72848. [CrossRef]
37. Mallozzi, P.; Incer, I.; Nuzzo, P.; Sangiovanni-Vincentelli, A. Contract-Based Specification Refinement and Repair for Mission Planning. In Proceedings of the 2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE), Victoria, Australia, 14–15 May 2023. [CrossRef]
38. Mallozzi, P.; Nuzzo, P.; Pelliccione, A.; Schneider, A. CROME: Contract-Based Robotic Mission Specification. In Proceedings of the 2020 ACM-IEEE 18th International Symposium on Formal Methods and Models for System Design (MEMOCODE), Jaipur, India, 2–4 December 2020. [CrossRef]
39. Cimatti, A.; Tonetta, S. A Property-Based Proof System for Contract-Based Design. In Proceedings of the 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Izmir, Turkey, 5–8 September 2012. [CrossRef]
40. Saoud, A.; Girard, A.; Fribourg, L. Assume–Guarantee Contracts for Continuous-Time Systems. *Automatica* **2021**, *134*, 109910. [CrossRef]
41. Nuzzo, P.; Lora, M.; Feldman, Y.A.; Sangiovanni-Vincentelli, A.L. CHASE: Contract-based Requirement Engineering for Cyber-Physical System Design. In Proceedings of the 2018 Design, Automation and Test in Europe (DATE), Dresden, Germany, 19–23 March 2018. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.