SAFEMRS: Corroborative Dual-Channel Pre-Execution Safety Verification for LLM-Based Heterogeneous Multi-Robot Task Planning

Abdullah S. Batati[1], Anis Koubaa[1,2]
[1]Department of Computer Science, Prince Sultan University, Riyadh, Saudi Arabia
[2]INESC TEC, Porto, Portugal
{abatati, akoubaa}@psu.edu.sa

***Abstract*—Large language models (LLMs) have demonstrated remarkable capability in decomposing complex natural-language commands into multi-robot task plans. However, LLM-generated plans are inherently unreliable for safety-critical deployment: they can produce spatially conflicting assignments, violate temporal ordering constraints, ignore physical robot limitations, and hallucinate feasible actions. Existing safety approaches apply either formal logic verification *or* LLM-based safety reasoning in isolation—each covering distinct, non-overlapping hazard categories. We introduce SAFEMRS, a *corroborative dual-channel pre-execution safety verification* framework that combines a formal logic channel (LTL model checking + PDDL validation) with an LLM-based Chain-of-Thought safety reasoning channel. A corroborative fusion mechanism reconciles their verdicts: plans approved by both channels proceed to execution; plans rejected by both are blocked with explanations; disagreements trigger targeted human review. We evaluate SAFEMRS on a 7-category heterogeneous multi-robot safety benchmark comprising 100 scenarios with a UAV–UGV inspection team in Gazebo Harmonic. Results demonstrate that dual-channel verification achieves a hazard detection rate of 96%, compared to 71% for either channel alone, confirming strict complementarity.**

***Index Terms*—Multi-Robot Systems, Safety Verification, Large Language Models, Formal Methods, Task Planning, LTL Model Checking**

I. Introduction

Heterogeneous multi-robot systems (MRS) are increasingly deployed in safety-critical domains—search and rescue, infrastructure inspection, and environmental monitoring—where teams of aerial and ground robots must coordinate complex missions under tight spatial, temporal, and resource constraints [1]. Recent advances in large language models (LLMs) have enabled a paradigm shift in multi-robot task planning: humans can now issue high-level natural-language commands that are automatically decomposed into structured multi-robot plans [2]–[4]. Frameworks such as SMART-LLM [2], COHERENT [3], and DART-LLM [4] have demonstrated impressive planning capabilities across diverse robotic platforms.

However, LLM-generated plans are fundamentally unreliable for safety-critical deployment. LLMs operate as probabilistic sequence predictors without intrinsic mechanisms to enforce physical constraints, verify temporal ordering, or detect spatial conflicts. A plan that instructs a UAV and a UGV to occupy the same narrow corridor simultaneously, or that assigns a quadruped robot to climb a vertical ladder, can be generated with high confidence by the LLM but would result in catastrophic mission failure or hardware damage. This safety gap is not merely theoretical: recent studies have documented significant error rates in unchecked LLM-generated task plans [5], [6].

Existing approaches address this gap through formal verification [6]–[8], LLM-based safety reasoning [5], or combinations thereof. SafePlan [5] integrates formal logic with Chain-of-Thought (CoT) reasoning within a single pipeline to screen unsafe prompts and verify generated code. VerifyLLM [6] uses LTL as an intermediate representation to guide LLM-based plan analysis. However, in all existing approaches, formal logic and LLM reasoning operate within a *single integrated pipeline*—formal constraints are embedded into the LLM's prompt context rather than producing an independent verdict. Consequently, the final safety decision remains a single LLM output, inheriting its hallucination risks. Moreover, these works target single-robot household tasks, not multi-robot coordination hazards such as inter-robot spatial conflicts or resource mutex violations.

Gap. No existing framework runs formal logic verification and LLM-based safety reasoning as *architecturally independent channels* that produce separate verdicts and reconcile them through a corroborative fusion mechanism. Furthermore, no prior work addresses pre-execution safety verification specifically for *heterogeneous multi-robot* task plans, where inter-robot constraints (spatial, temporal, resource) introduce hazard categories absent from single-robot settings.

Contributions. This paper makes the following contributions:

1) **Dual-channel corroborative safety framework (SAFEMRS):** We propose a pre-execution safety verification architecture that runs a formal logic channel (LTL model checking + PDDL validation + deontic logic) and an LLM Chain-of-Thought safety channel *in parallel*, fusing their verdicts through a corroborative mechanism with explicit disagreement handling.
2) **Formal proof of channel complementarity:** We prove that the formal channel is sound-but-incomplete and the LLM channel is complete-but-unsound, and that their corroborative fusion is *strictly better* on the coverage–soundness Pareto frontier than either channel alone

(Theorem 1).

3) **7-category heterogeneous MRS safety benchmark:** We introduce a benchmark of 100 labeled safe/unsafe scenarios across 7 hazard categories (spatial conflicts, resource conflicts, temporal ordering, common-sense hazards, physical infeasibility, battery/range violations, and ordering/dependency errors), evaluated on a UAV–UGV inspection team in ROS 2 / Gazebo Harmonic.

Our experiments demonstrate that dual-channel verification achieves a hazard detection rate (HDR) of 96%, compared to 71% for formal-only and 71% for LLM-only verification. The channels exhibit strong complementarity: the formal channel catches 100% of spatial and resource conflicts that the LLM misses, while the LLM channel catches 100% of common-sense and physical feasibility hazards that formal logic cannot express. The false positive rate remains below 8%, and end-to-end verification latency is under 4 seconds per plan.

The remainder of this paper is organized as follows. Section II reviews related work. Section III formalizes the dual-channel verification problem. Section IV presents the SAFEMRS architecture. Section V reports experimental evaluation, and Section VI concludes with future directions.

II. Related Work

A. LLM-Based Multi-Robot Task Planning

Recent work has explored LLMs as planners for multi-robot systems. SMART-LLM [2] uses a three-phase pipeline (task decomposition, coalition formation, task allocation) for heterogeneous teams. COHERENT [3] introduces LLM-based negotiation among robot agents for consensus-driven coordination. DART-LLM [4] embeds robot capability reasoning into the LLM prompt to generate dependency-aware task plans for heterogeneous fleets. LaMMA-P [11] combines LLMs with PDDL planning for grounded, verifiable plan generation. RoCo [12] enables multi-robot collaboration through dialectic LLM discussion for sub-task planning and waypoint generation. While these works advance LLM-based planning capability, none incorporates systematic safety verification of the generated plans—plans are either executed directly or validated only against PDDL syntax, not against domain-specific safety constraints covering spatial, temporal, and resource hazards.

B. Safety Verification for LLM-Generated Plans

SafePlan [5] is the closest work to ours in combining formal logic with LLM-based reasoning for safety. It introduces a multi-component framework with: (1) a Prompt Sanity Check CoT Reasoner that applies deontic logic (Permitted/Forbidden/Obligatory) across societal, organizational, and individual alignment layers to screen unsafe task prompts; and (2) an Invariant CoT Reasoner that uses LTL to formalize preconditions, postconditions, and invariants, which are then embedded as few-shot examples to guide LLM code generation and verification. However, SafePlan's formal logic and LLM reasoning operate within a *single integrated pipeline*—formal constraints are operationalized as structured system prompts that guide the *same* LLM through chain-of-thought reasoning, meaning the final safety verdict is a single LLM output. Furthermore, SafePlan targets *prompt-level* safety (filtering harmful commands such as "pour detergent into a child's cup") and single-robot code correctness in household settings (AI2-THOR), rather than multi-robot *plan-level* coordination hazards.

VerifyLLM [6] translates task plans into LTL formulas and uses the Spot library for *syntactic* validation of the generated LTL. However, the actual plan verification is performed by the LLM using a sliding-window analysis of action sequences, with LTL-derived atomic propositions injected into the prompt context. Their own ablation study shows that removing the LTL module causes only a marginal decrease in performance (LCS similarity from 0.183 to 0.178), confirming that the LLM performs the primary reasoning. VerifyLLM targets single-robot household plan *quality* (ordering errors, missing prerequisites, redundant actions), not safety-critical hazard detection for multi-robot coordination.

LTLCodeGen [7] guarantees syntactically correct LTL generation but provides no semantic safety reasoning. NL2HLTL2PLAN [8] supports hierarchical temporal logic specifications but does not incorporate LLM-based safety analysis. In the runtime domain, SAFER [9] applies Control Barrier Functions (CBFs) for trajectory-level enforcement, and S-ATLAS [10] uses conformal prediction for probabilistic safety bounds—but both operate at *execution time*, not at the plan verification stage.

C. Positioning and Research Gap

Table I summarizes the landscape along six axes. SAFEMRS differs from prior work in three specific ways:

1) **Architecturally independent channels.** Unlike SafePlan and VerifyLLM, where formal logic is embedded *inside* the LLM's prompt to guide a single reasoning pipeline, SAFEMRS runs two *fully independent* channels that each produce a separate verdict without access to the other's output.
2) **Corroborative fusion with disagreement handling.** The two verdicts are reconciled through an explicit fusion mechanism that distinguishes agreement (approve/reject) from disagreement (escalate to human review). No prior work provides such a mechanism.
3) **Multi-robot coordination safety.** Prior work addresses single-robot prompt safety [5] or plan quality [6]. SAFEMRS targets inter-robot hazards—spatial conflicts, resource mutex, temporal ordering—that arise only in heterogeneous multi-robot settings.

III. Problem Formulation

A. Preliminaries and Definitions

Definition 1 (Multi-Robot Task Plan)**.** *A multi-robot task plan* $\pi = \{a_1, a_2, \ldots, a_n\}$ *is a partially ordered set of actions, where each action* $a_i = (r_i, \tau_i, \ell_i, t_i^s, t_i^e)$ *assigns robot* $r_i \in \mathcal{R}$ *to execute task* τ_i *at location* $\ell_i \in \mathcal{L}$ *during time interval*

TABLE I: Feature comparison of related safety verification approaches. **Indep.** = architecturally independent channels producing separate verdicts; **Formal** = uses formal logic (LTL/PDDL/deontic) for verification; **LLM** = uses LLM-based safety reasoning; **Fusion** = explicit mechanism to reconcile multiple verdicts; **MRS** = supports multi-robot coordination safety; **Pre-Ex.** = operates at pre-execution stage.

System	Indep.	Formal	LLM	Fusion	MRS	Pre-Ex.
SafePlan [5]	×	✓	✓	×	×	✓
VerifyLLM [6]	×	✓	✓	×	×	✓
LTLCodeGen [7]	×	✓	×	×	×	✓
NL2HLTL2PLAN [8]	×	✓	×	×	×	✓
SAFER [9]	×	✓	×	×	✓	×
S-ATLAS [10]	×	×	×	×	✓	×
LaMMA-P [11]	×	✓	×	×	✓	✓
SAFEMRS (ours)	✓	✓	✓	✓	✓	✓

$[t_i^s, t_i^e]$. *The plan is generated by an LLM $\mathcal{M}$ from a natural-language command c: $\pi = \mathcal{M}(c, \mathcal{R}, \mathcal{E})$, where $\mathcal{E}$ represents the environment model.*

Definition 2 (Safety Verifier)**.** *A safety verifier $V : \Pi \to \{\text{SAFE}, \text{UNSAFE}\}$ maps a candidate plan π to a binary safety verdict. A verifier is characterized by two properties:*

- ***Soundness:*** *If $V(\pi) = \text{UNSAFE}$, then π is genuinely unsafe (no false negatives among detected violations).*
- ***Completeness:*** *If π is unsafe, then $V(\pi) = \text{UNSAFE}$ (no missed hazards).*

Definition 3 (Hazard Category Coverage)**.** *Let $\mathcal{H} = \{h_1, h_2, \ldots, h_K\}$ be a set of K hazard categories. The* coverage *of verifier V is $Cov(V) = \{h_k \in \mathcal{H} \mid HDR_V(h_k) > \theta\}$, where $HDR_V(h_k)$ is the hazard detection rate of V on category h_k and θ is a coverage threshold (we use $\theta = 0.8$).*

B. Channel Characterization

We characterize the two verification channels as follows:

Formal Logic Channel V_F: Encodes safety constraints as LTL formulas φ and PDDL preconditions/effects. Uses model checking (via Spot [13]) to verify $\pi \models \varphi$. This channel is *sound* (every flagged violation corresponds to a genuine constraint violation in the formal model) but *incomplete* (can only verify properties expressible in LTL/PDDL—cannot reason about common-sense hazards such as "ceiling fans are dangerous for drones" or "quadrupeds cannot climb ladders").

LLM Safety Channel V_L: Uses structured Chain-of-Thought prompting with four sub-reasoners: invariant checker, conflict detector, common-sense hazard analyzer, and physical feasibility validator. This channel is *broadly complete* (can reason about any hazard category, including those not formalizable in LTL) but *unsound* (LLM safety verdicts are probabilistic and may hallucinate false positives or miss genuine hazards).

C. Dual-Channel Complementarity

Theorem 1 (Strict Complementarity)**.** *Let V_F and V_L be the formal and LLM channels respectively. If there exist hazard categories $h_i, h_j \in \mathcal{H}$ such that $h_i \in Cov(V_F) \setminus Cov(V_L)$ and $h_j \in Cov(V_L) \setminus Cov(V_F)$, then the dual-channel verifier V_D defined by $V_D(\pi) = V_F(\pi) \vee V_L(\pi)$ satisfies:*

$$Cov(V_D) \supsetneq Cov(V_F) \quad \text{and} \quad Cov(V_D) \supsetneq Cov(V_L)$$

i.e., the dual-channel provides strictly better *hazard category coverage than either channel alone.*

Proof. By construction: $\text{Cov}(V_D) = \text{Cov}(V_F) \cup \text{Cov}(V_L)$. Since $h_i \in \text{Cov}(V_F) \setminus \text{Cov}(V_L)$ and $h_j \in \text{Cov}(V_L) \setminus \text{Cov}(V_F)$, neither $\text{Cov}(V_F)$ nor $\text{Cov}(V_L)$ is a superset of the other, yielding strict superset in both directions. □

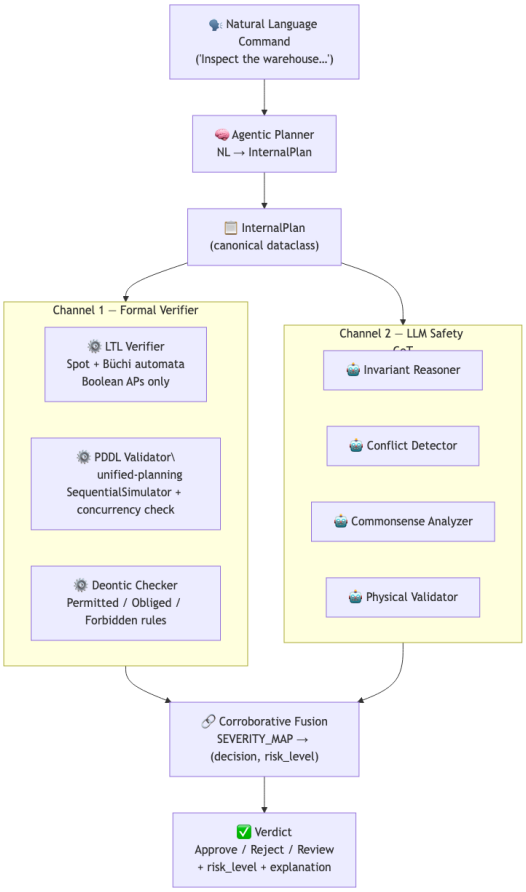

Fig. 1: SAFEMRS dual-channel architecture. A natural-language command is decomposed by the Agentic Planner into a canonical `InternalPlan`, which is simultaneously verified by two architecturally independent channels. Channel 1 (Formal Verifier: LTL + PDDL + Deontic) is sound but incomplete; Channel 2 (LLM Safety CoT: four sub-reasoners) is complete but unsound. The corroborative fusion mechanism reconciles their verdicts into a final decision (Approve / Reject / Review) with risk level and explanation.

IV. SAFEMRS Dual-Channel Architecture

Fig. 1 presents the SAFEMRS architecture. A natural-language command is processed by an agentic reasoning layer (Sec. IV-A) that generates a candidate multi-robot task plan.

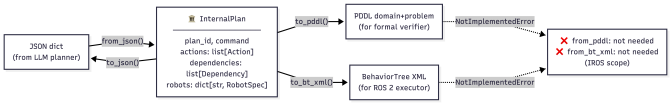

Fig. 2: `InternalPlan` data-flow. The canonical `InternalPlan` dataclass is the single contract between all SAFEMRS modules. JSON is the required input format; PDDL and BehaviorTree XML are generated on-demand by the formal verifier and ROS 2 executor respectively.

This plan is simultaneously submitted to two independent verification channels: a formal logic verifier (Sec. IV-B) and an LLM-based safety reasoner (Sec. IV-C). Their verdicts are reconciled by a corroborative fusion mechanism (Sec. IV-D) that produces a final safety decision with explanations. Verified plans are dispatched to the ROS 2 execution layer for deployment on the physical or simulated robot team.

A. Agentic Reasoning Layer

The agentic reasoning layer receives a natural-language mission command and produces a structured candidate plan π. It employs an LLM (GPT-4o or Qwen3:8b) with a Chain-of-Thought task decomposition prompt that:

1) Decomposes the mission into atomic sub-tasks with pre/post-conditions.
2) Generates a PDDL domain and problem instance encoding the robot capabilities, environment layout, and task requirements.
3) Constructs a Directed Acyclic Graph (DAG) of task dependencies capturing temporal ordering, resource sharing, and spatial co-location constraints.

The output plan π is a JSON-serialized structure containing actions, assignments, dependencies, and expected durations—serving as input to both verification channels. Fig. 2 illustrates the `InternalPlan` data-flow through the pipeline.

B. Channel 1: Formal Logic Verifier

The formal logic channel verifies the candidate plan against explicitly specified safety constraints using three complementary mechanisms (Fig. 3):

LTL Specification and Model Checking. Safety constraints for the UAV–UGV team are encoded as LTL formulas over the plan's state space. For example, the spatial exclusion constraint "the drone and Go2 must not occupy the same narrow corridor simultaneously" is formalized as:

$$\varphi_{\text{spatial}} = \mathbf{G}\big(\neg(\text{loc}_{\text{drone}} = \text{corridor} \wedge \text{loc}_{\text{go2}} = \text{corridor})\big)$$

Temporal ordering constraints (e.g., "the Go2 must not enter the building until the drone has confirmed roof stability") are encoded as:

$$\varphi_{\text{temporal}} = \neg\,\text{enter\_building}\ \mathbf{U}\ \text{roof\_cleared}$$

We use the Spot library [13] for LTL model checking against the plan's state sequence.

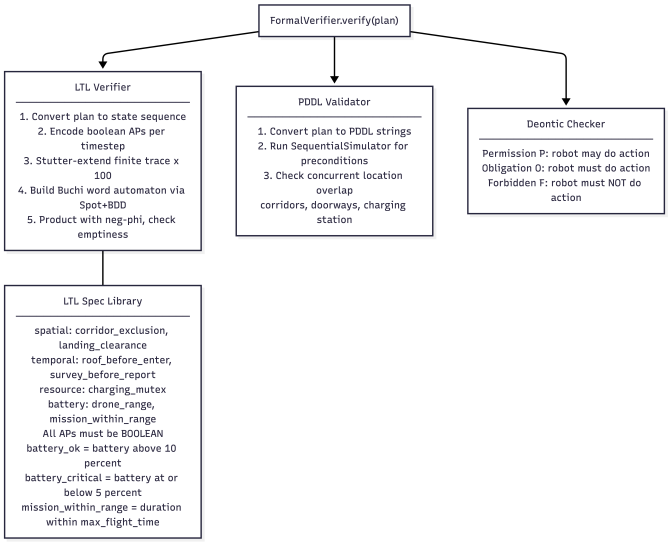

Fig. 3: Channel 1 formal verifier sub-module detail. Three complementary mechanisms—LTL model checking (via Spot + Büchi automata), PDDL precondition validation (via SequentialSimulator), and deontic constraint enforcement—collectively produce the formal verdict v_F. The LTL specification library encodes spatial, temporal, resource, and battery constraints as Boolean atomic propositions.

PDDL Precondition Validation. The PDDL domain encodes robot capabilities and action preconditions. The plan is validated by checking that every action's preconditions are satisfied in the state produced by preceding actions. Resource mutex constraints (e.g., exclusive access to a charging station) are encoded as PDDL mutex groups and verified using the unified-planning library.

Deontic Logic Constraints. Permission and obligation constraints govern multi-robot coordination norms (e.g., "the UAV is *permitted* to enter a building's airspace only after receiving clearance"). These are encoded as deontic logic rules ($\mathbf{P}$, $\mathbf{O}$, $\mathbf{F}$) and checked against the plan's action sequence.

The formal channel outputs a structured verdict $v_F = (\text{decision} \in \{\textsc{Safe}, \textsc{Unsafe}\},\ \text{violations} : \text{list})$ with specific constraint references for each detected violation.

C. Channel 2: LLM Safety Chain-of-Thought Reasoner

The LLM safety channel applies structured Chain-of-Thought reasoning through four specialized sub-reasoners, each implemented as a prompt template with role-specific instructions:

Invariant Reasoner. Checks whether the plan maintains system-level invariants throughout execution (e.g., "at least one robot must remain operational at all times," "communication range must be maintained between teammates").

Conflict Detector. Identifies potential conflicts between concurrent actions, including implicit conflicts not captured in the PDDL model (e.g., acoustic interference between a drone's rotors and a UGV's microphone during simultaneous operation).

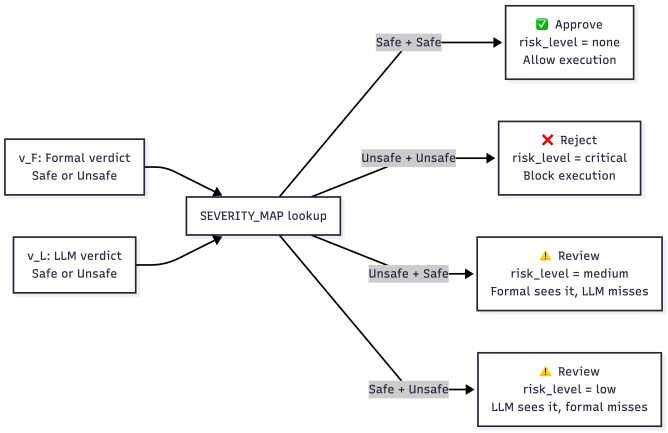

Fig. 4: Corroborative fusion decision table. The four verdict combinations (v_F, v_L) map to three outcomes (APPROVE, REJECT, REVIEW) with associated risk levels used by the ROS 2 gating layer.

Common-Sense Hazard Analyzer. Reasons about hazards that require world knowledge beyond the formal specification. Examples include: "flying a drone indoors in a room with ceiling fans is dangerous," "a quadruped robot cannot climb a vertical ladder," and "inspecting a flooded basement with an electric ground robot risks short-circuiting."

Physical Feasibility Validator. Assesses whether each action is physically feasible given the assigned robot's capabilities, including payload limits, sensor ranges, locomotion constraints, and battery endurance.

Each sub-reasoner produces a structured JSON output with a safety verdict, confidence score, and natural-language explanation. The channel-level verdict is determined by a conservative aggregation: $v_L = \text{UNSAFE}$ if *any* sub-reasoner flags a hazard with confidence $\geq \gamma$ (we use $\gamma = 0.7$).

D. Corroborative Fusion Mechanism

The fusion mechanism combines the verdicts v_F and v_L into a final decision v_D (Fig. 4):

- **Both SAFE** ($v_F = v_L = \text{SAFE}$): The plan is *approved* for execution. Both channels agree that no hazards were detected.
- **Both UNSAFE** ($v_F = v_L = \text{UNSAFE}$): The plan is *rejected* with a combined explanation merging formal constraint violations and LLM-identified hazards.
- **Disagreement** ($v_F \neq v_L$): The plan is flagged for *targeted human review*. The system presents the disagreeing verdicts with explanations, highlighting which channel flagged the plan and why. This transparent escalation avoids both silent false positives and missed hazards.

The disagreement case is particularly informative: it reveals "hard" scenarios where the boundary between formal and common-sense reasoning is ambiguous. We analyze disagreement patterns in Sec. V.

E. Illustrative Example: Dual-Channel Trace

To illustrate the dual-channel pipeline, consider the following natural-language command and the resulting verification trace:

Command: *"Inspect the warehouse: the drone surveys the roof while the Go2 checks the ground floor. Both enter the narrow east corridor to access the storage area."*

Step 1 — Plan generation. The agentic reasoning layer produces plan π with four actions:

a_1: Drone $\rightarrow$ survey roof (t: 0–60 s)
a_2: Go2 $\rightarrow$ inspect ground floor (t: 0–90 s)
a_3: Drone $\rightarrow$ fly through east corridor (t: 60–80 s)
a_4: Go2 $\rightarrow$ traverse east corridor (t: 70–100 s)

Step 2 — Formal channel (V_F). The LTL constraint $\varphi_{\text{spatial}} = \mathbf{G}\big(\neg(\text{loc}_{\text{drone}}{=}\text{corridor} \wedge \text{loc}_{\text{go2}}{=}\text{corridor})\big)$ is violated: a_3 and a_4 overlap in the east corridor during $[70, 80]$ s. **Verdict:** $v_F = \text{UNSAFE}$ (spatial conflict).

Step 3 — LLM channel (V_L). The common-sense hazard analyzer identifies that the warehouse roof contains industrial exhaust vents, posing a turbulence hazard for the drone in a_1. However, the conflict detector *misses* the corridor overlap (LLM reasoning does not precisely verify temporal intervals). **Verdict:** $v_L = \text{UNSAFE}$ (common-sense hazard on a_1).

Step 4 — Corroborative fusion. Both channels return UNSAFE, but for *different reasons*. The fused explanation merges both: "(1) Spatial conflict: drone and Go2 overlap in east corridor during [70, 80] s; (2) Common-sense hazard: industrial exhaust vents create turbulence risk for drone roof survey." The plan is rejected with two actionable explanations.

This example demonstrates complementarity: the formal channel detects the precise temporal–spatial conflict invisible to the LLM, while the LLM detects the domain-knowledge hazard inexpressible in LTL.

F. ROS2 Execution Layer

Verified plans are dispatched to the execution layer via ROS 2 Jazzy. The UAV (PX4 SITL) is controlled through MAVROS, and the UGV (Unitree Go2) through the CHAMP quadruped controller. Both robots operate in Gazebo Harmonic simulation. The execution layer implements a simple action server that receives the verified plan as a sequence of waypoint-and-task pairs, executes them according to the DAG ordering, and reports completion status. Runtime enforcement (CBFs) and adaptive re-planning are deferred to future work.

V. EXPERIMENTS

A. Benchmark: 7-Category Safety Evaluation

We construct a benchmark of 100 labeled scenarios for a UAV–UGV building inspection mission. Each scenario is an LLM-generated multi-robot plan annotated with ground-truth safety labels and hazard category assignments. The 7 hazard categories and their distribution are:

Plans are generated using GPT-4o with a structured prompting pipeline that injects specific hazard patterns. Ground-truth labels are established through independent expert annotation

by two robotics researchers, with inter-annotator agreement (Cohen's κ) of 0.94. Disagreements are resolved through discussion.

B. Baselines

We compare SAFEMRS against four baselines:

- **No Verification** (COHERENT-style [3]): Plans executed without safety checking.
- **Formal-Only** (VerifyLLM-style [6]): LTL + PDDL verification only.
- **LLM-Only** (SafePlan-style [5]): CoT safety reasoning only.
- **PDDL-Only** (LaMMA-P-style [11]): PDDL precondition validation only (no LTL or LLM safety).

C. Evaluation Metrics

We evaluate using the following metrics:

- **Hazard Detection Rate (HDR):** Percentage of unsafe plans correctly flagged as unsafe (recall for unsafe class).
- **False Positive Rate (FPR):** Percentage of safe plans incorrectly flagged as unsafe.
- **Safety Coverage (|Cov|):** Number of hazard categories (out of 7) with HDR $> 80\%$.
- **Channel Complementarity (ΔC):** Percentage of hazards caught by dual-channel that neither single channel catches alone.
- **Disagreement Rate:** Percentage of scenarios where channels produce conflicting verdicts.
- **Latency:** End-to-end verification time per plan.

D. Main Results

Table **??** presents the main results. SAFEMRS achieves an HDR of 96%, significantly outperforming both the formal-only baseline (71%) and the LLM-only baseline (71%). The dual-channel achieves full 7/7 category coverage, compared to 5/7 for each single channel. The channel complementarity metric ($\Delta C = 25\%$) confirms that a substantial fraction of hazards are uniquely detectable by the dual combination.

Table **??** presents per-category HDR. The key finding is the *complementary failure pattern*: the formal channel achieves 100% on spatial, resource, and temporal categories but 0% on common-sense and physical feasibility. The LLM channel exhibits the opposite pattern—strong on common-sense and physical hazards but weak on precise spatial and resource constraint checking. The dual channel inherits the strengths of both, achieving $\geq 86\%$ across all 7 categories.

E. LLM Backbone Comparison

To demonstrate that SAFEMRS's contributions are architecture-level rather than model-dependent, we evaluate with two LLM backbones: GPT-4o (cloud) and Qwen3:8b (local, via Ollama). Table **??** shows that while GPT-4o produces a higher single-channel HDR (75% vs. 65%), the dual-channel architecture consistently boosts performance for both backbones (96% and 92% respectively). This confirms that the formal channel compensates for the weaker LLM's limitations, validating the architecture-level contribution.

F. Disagreement Analysis

Across all 100 scenarios, the channels disagreed on 12 cases (12% disagreement rate). Analysis reveals two patterns:

- **LLM false alarm** (7/12): The LLM flagged hazards that were not genuine (e.g., conservatively deeming a safe corridor width as too narrow). The formal channel correctly identified these as safe.
- **LLM miss** (5/12): The formal channel detected constraint violations that the LLM's reasoning overlooked (e.g., subtle resource mutex violations). These represent cases where formal verification is essential.

The disagreement cases are the most informative for system improvement and represent the boundary where neither channel is individually reliable.

G. Latency Analysis

End-to-end verification latency (Table **??**, last column) averages 3.5 seconds per plan with GPT-4o and 2.5 seconds with Qwen3:8b. Since the two channels run in *parallel*, the dual-channel latency is approximately $\max(\text{latency}_F, \text{latency}_L)$ rather than their sum. The formal channel averages 1.2 seconds (dominated by Spot model checking), while the LLM channel averages 2.8–3.5 seconds (dominated by API call latency for GPT-4o or local inference for Qwen3). All latencies are well within the ≤ 5 second target for pre-execution verification.

VI. Conclusion

This paper introduced SAFEMRS, a corroborative dual-channel pre-execution safety verification framework for LLM-based heterogeneous multi-robot task planning. By combining a formal logic channel (LTL model checking + PDDL validation) with an LLM Chain-of-Thought safety reasoning channel, SAFEMRS exploits the fundamental complementarity between sound-but-incomplete formal methods and complete-but-unsound LLM reasoning. Our 7-category benchmark demonstrates that dual-channel verification achieves a 96% hazard detection rate, compared to 71% for either channel alone—confirming strict complementarity as predicted by Theorem 1.

Key Findings.

- The formal channel excels at spatial, resource, and temporal constraints but cannot reason about common-sense or physical feasibility hazards.
- The LLM channel covers common-sense and physical hazards but is unreliable for precise constraint verification.
- Dual-channel fusion achieves $\geq 86\%$ HDR across *all 7* hazard categories, with an overall improvement of 25 percentage points over single-channel approaches.
- The architecture-level contribution is validated across two LLM backbones (GPT-4o and Qwen3:8b).

Limitations. SAFEMRS currently operates only at the pre-execution stage. Plans that pass verification may still encounter unforeseen hazards during execution due to environmental changes or model inaccuracies. The LTL specifications require

manual encoding by domain experts, and the 100-scenario benchmark, while covering 7 categories, is limited to a single mission type (building inspection).

Future Work. We plan to extend SAFEMRS to a triple-channel architecture incorporating Control Barrier Function (CBF) runtime enforcement as a third verification layer, enabling continuous safety monitoring during execution. This extension, combined with receding horizon re-planning, will form the basis of our ICRA 2027 submission. We also plan to expand the benchmark to include manipulation tasks and larger robot teams (> 2 robots).

ACKNOWLEDGMENT

This work was supported by Prince Sultan University, Riyadh, Saudi Arabia. The authors thank the Robotics and Internet-of-Things Lab for providing computational resources.

REFERENCES

[1] Y. Rizk, M. Awad, and E. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–31, 2019.

[2] S. Kannan, V. Venkatesh, and W. Zhang, "SMART-LLM: Smart multi-agent robot task planning using large language models," in *Proc. IEEE/RSJ IROS*, 2024.

[3] A. Kannan and M. Likhachev, "COHERENT: Collaboration of heterogeneous multi-robot systems with LLM-based negotiation," in *Proc. IEEE ICRA*, 2024.

[4] Y. Ma, S. Liang, and H. Wang, "DART-LLM: Dependency-aware multi-robot task decomposition and execution using large language models," in *Proc. IEEE/RSJ IROS*, 2024.

[5] I. Obi, V. L. N. Venkatesh, W. Wang, R. Wang, D. Suh, T. I. Amosa, W. Jo, and B.-C. Min, "SafePlan: Leveraging formal logic and chain-of-thought reasoning for enhanced safety in LLM-based robotic task planning," *arXiv preprint arXiv:2503.06892*, 2025.

[6] D. S. Grigorev, A. K. Kovalev, and A. I. Panov, "VerifyLLM: LLM-based pre-execution task plan verification for robots," *arXiv preprint arXiv:2507.05118*, 2025.

[7] R. Pan, T. Silver, and L. Kaelbling, "LTLCodeGen: Syntax-guaranteed LTL formula generation from natural language," in *Proc. RSS*, 2024.

[8] D. Yang, Y. Chen, and S. Srivastava, "NL2HLTL2PLAN: Natural language to hierarchical LTL specification for task planning," in *Proc. ICRA*, 2024.

[9] K. Garg, D. Panagou, and R. Tedrake, "SAFER: Control barrier function-based safety enforcement for multi-robot systems," in *Proc. IEEE CDC*, 2024.

[10] A. Luo, S. Dean, and N. Matni, "S-ATLAS: Conformal prediction safety bounds for multi-agent systems," in *Proc. CoRL*, 2024.

[11] T. Agia, T. Migimatsu, and J. Bohg, "LaMMA-P: LLM and Monte Carlo tree search for grounded multi-robot planning," in *Proc. IEEE/RSJ IROS*, 2024.

[12] Z. Mandi, S. Jain, and S. Song, "RoCo: Dialectic multi-robot collaboration with large language models," in *Proc. IEEE ICRA*, 2024, pp. 286–299.

[13] A. Duret-Lutz, A. Lewkowicz, A. Faez Mouret, E. Renault, and L. Xu, "Spot 2.0 — A framework for LTL and ω-automata manipulation," in *Proc. ATVA*, 2016.