maPO: An Ontology for Multi-Agent Path Finding and Its Usage for Explaining Planner Behaviour

Anonymous submission

Abstract

As multi-agent systems become more autonomous, particularly in complex coordination tasks like Multi-Agent Path Finding (MAPF), the need for transparent and interpretable decision-making becomes critical. Although execution traces from MAPF algorithms provide rich diagnostic insight, existing explainability methods like visual segmentation of trace snapshots and logic-based "why" queries address individual modalities but remain fragmented. We introduce the *Multi-Agent Planning Ontology (maPO)*, a unified semantic schema that turns raw MAPF traces into a single knowledge graph, formalizing segmentation snapshots, conflict alerts, and replanning strategies. Our log-to-graph pipeline ingests planner outputs as ontology instances, and SPARQL queries produce contrastive and logical explanations. A user study (N=25) confirms the effectiveness of our approach, showing that our generated explanations are preferred over raw data 94% of the time ($p < .001$) and are rated as significantly clearer. Our contributions are: (1) the MA Planning Ontology schema, (2) a log-to-graph transformation pipeline for SPARQL-based explanation generation, and (3) an empirical validation of the explanation generation framework.

1 Introduction

Coordinating multiple autonomous agents to reach individual goals without collisions is a foundational challenge in robotics and AI. Multi-Agent Path Finding (MAPF) formalizes this problem on a shared graph and is known to be NP-hard in its general form (Sharon et al. 2013; Ren et al. 2025). Modern planners such as Conflict-Based Search (CBS) and its improved variants (Sharon et al. 2015; Boyarski et al. 2015), as well as reinforcement-learning approaches like PRIMAL (Sartoretti et al. 2019; Damani et al. 2021), achieve high performance but offer little transparency into their decision processes.

Recent research on MAPF explainability has explored several complementary directions. A visual segmentation approach (Almagor and Lahijanian 2020) decomposes a joint plan into a minimal sequence of non-conflicting snapshots for easy human verification. Algorithmic integration of explainability appears in (Kottinger, Almagor, and Lahijanian 2022), which extends CBS to favor solutions admitting short segmentation-based explanations. A user-driven taxonomy (Brandao et al. 2022) identifies the explanation types stakeholders need (e.g. infeasibility, suboptimality, agent delays). Logic-based frameworks such as (Bogatarkan 2021) use Answer Set Programming to answer "why" and "why not" queries directly from the planning model. Despite these advances, there is no unified framework that both formalizes MAPF concepts and supports diverse explanation modalities at scale.

In this paper, we present the *Multi-Agent Planning Ontology (maPO)*, an extension of the standard Planning Ontology that captures MAPF-specific constructs, including agent properties, collision events, conflict alerts, and replanning strategies, and the causal relations among them. By transforming execution traces into a semantic knowledge graph, our ontology enables on-demand SPARQL queries without modifying the underlying path-planning algorithms. We demonstrate that our approach imposes negligible overhead, aligns with user needs identified in prior taxonomies (Brandao et al. 2022), and generalizes across MAPF variants. Our contributions in this paper are: **(1)** the *maPO* schema, **(2)** a SPARQL-based explanation generation framework utilizing *maPO* schema, and **(3)** a user study demonstrating the effectiveness of our framework. The remainder of the paper is organized as follows. Section 2 surveys the MAPF algorithms, explanation generation methods, and existing ontologies for autonomous systems; Section 3 introduces the (*maPO*) schema; Section 4 details our SPARQL-based explanation framework; Section 5 reports the user study results; and Section 6 concludes and outlines future work.

2 Background & Literature Review

MAPF Problem Formulation

Multi-Agent Path Finding is defined on an undirected graph $G = (V, E)$, where V represents grid cells (vertices) and E represents connections between adjacent cells (edges) (Wang et al. 2025). A team of n agents, $A = \{a_1, \ldots, a_n\}$, each with a unique start vertex $s_i \in V$ and goal vertex $g_i \in V$, must navigate this environment (Wang et al. 2023b). Time is discretized into steps $t = 0, 1, 2, \ldots$, and at each step, an agent may either move along an edge or wait at its current vertex (Wang et al. 2023b). An agent's path π_i is a sequence of vertices $(v_0^i, v_1^i, \ldots, v_{T_i}^i)$, where $v_0^i = s_i$ and $v_{T_i}^i = g_i$ (Wang et al. 2023b). A solution $\Pi = \{\pi_1, \ldots, \pi_n\}$ is collision-free if, for all distinct agents $i \neq j$ and all time

steps t: **Vertex-collision free**: $v_t^i \neq v_t^j$ (no two agents occupy the same vertex at the same time) (Wang et al. 2023b). **Edge-collision free**: $(v_t^i, v_{t+1}^i) \neq (v_{t+1}^j, v_t^j)$ (agents do not traverse the same edge in opposite directions simultaneously) (Wang et al. 2023b). Common efficiency objectives include minimizing the makespan (the time when the last agent reaches its goal), minimizing the sum of individual arrival times (sum-of-costs), or minimizing the total number of collisions encountered (Wang et al. 2023b). An important modeling choice is how agents behave after reaching their goals: in the "stay at target" setting, agents remain at their goal vertices (possibly blocking others), whereas in the "disappear at target" setting, agents are removed from the environment upon arrival (Sharon et al. 2015; Stern et al. 2019).

An Overview of MAPF Algorithms

A wide spectrum of algorithms has been developed to solve the MAPF problem, each embodying different trade-offs between solution optimality, computational scalability, and information requirements. Generically, the MAPF pipeline can be conceptualized in four stages: **S1** (initial agent planning), **S2** (collision detection), **S3** (collision resolution), and **S4** (agent replanning).

Centralized algorithms, such as Conflict-Based Search (CBS) (Sharon et al. 2015) and its variants like Improved CBS (ICBS) (Boyarski et al. 2015), operate with a global view of the environment. They systematically identify and resolve conflicts between agent paths, often guaranteeing optimal solutions with respect to cost or makespan. However, this guarantee comes at a high computational cost that grows with the number of agents and conflicts, and it requires that all agent information be available to a single planner. In contrast, **decentralized and distributed approaches** prioritize scalability by limiting the information available to each agent. These methods range from reinforcement learning policies, where agents learn to coordinate implicitly based on local observations (e.g., PRIMAL (Sartoretti et al. 2019; Damani et al. 2021)), to fully decentralized techniques that rely only on on-board sensing and learned rules with no communication at all (e.g., SCRIMP (Wang et al. 2023a)). While these methods scale to much larger teams, they often sacrifice optimality and may not guarantee completeness.

Hybrid frameworks aim to achieve the best of both worlds by combining fast, decentralized planning with a lightweight centralized coordinator for resolving complex conflicts. For instance, approaches like LNS2+RL (Li et al. 2022; Wang et al. 2025) use learned policies for local agent movement and a large-neighborhood search to repair global conflicts as they arise. This demand-driven coordination reduces communication overhead while maintaining high success rates.

Despite this algorithmic diversity, from systematic global search to learned local policies, our explanation framework remains universally applicable. By focusing on the *output* of the planning process rather than its internal mechanics, our ontology can provide consistent, structured explanations for any planner capable of producing a standardized execution trace, discussed in Section 4.

Ontologies for Autonomous Systems and Planning

The use of ontologies to formalize knowledge in robotics and autonomous systems is a well-established practice aimed at promoting interoperability, reusability, and formal reasoning. Foundational efforts like the Planning Ontology (PO) (Muppasani et al. 2024) provide a vocabulary for describing sequential plans and processes for the field of automated planning. In robotics, the IEEE standard Core Ontology for Robotics and Automation (CORA) offers a rich model for physical robots, their capabilities, and environments (Schlenoff et al. 2012). For modeling perception and interaction, the W3C standard SOSA/SSN ontology provides a vocabulary to describe sensors, observations, and the platforms that host them, which is critical for grounding agent perception in a formal structure (Janowicz et al. 2019).

Temporal and historical context is equally important. The W3C Time Ontology provides a standard for representing time instants and intervals (Pan and Hobbs 2006), while the PROV Ontology (PROV-O) offers a powerful, domain-agnostic framework for modeling provenance that is, the history and derivation of data and artifacts (Lebo et al. 2013). PROV-O is particularly relevant for explainability, as it can formally capture how a plan is revised or derived from another, creating a traceable, auditable record of the planning process. Our work builds upon these principles, reusing concepts from these established standards to ensure our ontology is both robust and interoperable.

Explainability in MAPF

As MAPF systems move into safety-critical and regulatory contexts, users and stakeholders demand not only correct but also understandable plans. Early work (Almagor and Lahijanian 2020) introduced a *plan-segmentation* explanation paradigm, in which a complex multi-agent execution trace is decomposed into a minimal sequence of collision-free snapshots that a human can quickly verify for safety. Building on this idea, (Kottinger, Almagor, and Lahijanian 2022) extended Conflict-Based Search to *prefer* solutions that admit short segmentation-based explanations, effectively embedding explainability constraints into the planner at minimal additional cost. Complementing these algorithmic advances, (Brandao et al. 2022) conducted an expert user study to derive a detailed taxonomy of explanation needs, such as plan infeasibility, suboptimality justifications, and agent wait-time clarifications, and recommended corresponding modalities (visual, textual, contrastive) for effective presentation. In parallel, (Bogatarkan 2021) demonstrated that a logic-based framework using Answer Set Programming can answer rich "why" and "why not" queries about MAPF solutions by reasoning over the same constraints that generate the plan.

Ontology-based representations offer a unified structure for all explanation modalities. By encoding agent states, path segments, conflict alerts, and replanning strategies, explanation requests, whether *visual* ("show me the collision-free segments"), *contrastive* ("why this path instead of that one?") or *logical* ("why was the plan not infeasible?"), can all be expressed as SPARQL queries over

the same knowledge graph. This approach eliminates the need for separate pipelines for visual segmentation and logical reasoning, leverages mature semantic-web tools for extension and maintenance, and ensures that new explanation forms (e.g., counterfactuals or temporal summaries) can be added simply by defining new ontology classes or query templates. To realize this, we introduce the *maPO*, which formalizes the conflict-resolution lifecycle in OWL and demonstrates how a single, coherent framework can generate rich, on-demand explanations across diverse MAPF scenarios.

3 Multi-Agent Planning Ontology

Building upon the foundational concepts of the Planning Ontology (Muppasani et al. 2024) described previously, we introduce the *maPO*, presented in Figure 1. This extension is specifically designed to address the unique complexities of multi-agent scenarios and to establish a formal, queryable knowledge base that supports on-demand explainability. To ensure interoperability and community acceptance, our *maPO* reuses concepts from established W3C and IEEE standards where appropriate. While the core categories of the base ontology are preserved, they are enhanced to model agent-centric information, inter-agent conflicts, and the procedural rationale behind conflict resolution. This structure transforms opaque execution trace data into a queryable knowledge graph, enabling the systematic generation of answers to complex explanatory competency questions.

Reuse of Standard Ontologies

To ground our ontology in established semantic standards, we reuse concepts from several widely adopted resources. This practice enhances interoperability and aligns our work with community best practices. Table 1 summarizes the key reused ontologies and their roles within our framework.

Table 1: Summary of Reused Ontologies

Standard	Reused Concept	Purpose in Framework
SOSA	`sosa:Platform`	Models agent as a sensor platform
CORA	`cora:Capability`	Defines agent capabilities
OWL-Time	`time:Instant`	Represents event times
PROV-O	`prov:wasDerivedFrom`	Links original to revised plans
RDF	`rdf:Seq`	Structures agent path sequences

Competency Questions

To ensure our ontology effectively supports explainability, we defined a set of competency questions (CQs) that guide its design and scope. These questions represent concrete explanatory needs that an analyst or end-user would have when trying to understand a multi-agent plan. The ontology must contain the necessary classes and properties to answer each of these questions via SPARQL queries. The following CQs were developed to address the specific challenges of multi-agent plan explanation:

- **C1:** Which `CollisionEvents` (including their time, type, location, and involved agents) were detected during planning?
- **C2:** For a given `CollisionEvent`, which agent(s) received a `ConflictAlert`?
- **C3:** What was an agent's original, conflict-unaware plan, and how does it compare to its final, resolved plan?
- **C4:** Why did a specific agent have to wait or reroute in its final plan?
- **C5:** For a given `ConflictAlert`, which `ReplanningStrategy` did the agent use?
- **C6:** What was the cost change associated with a revised `AgentSubPlan`?
- **C7:** Why was a particular agent (from a set of conflicting agents) chosen to be the one to replan? (i.e., what was the planner's `selectionRationale`?)
- **C8:** What is the final `JointPlan` after all conflicts are resolved, and what is its overall makespan?

Agent and State Representation

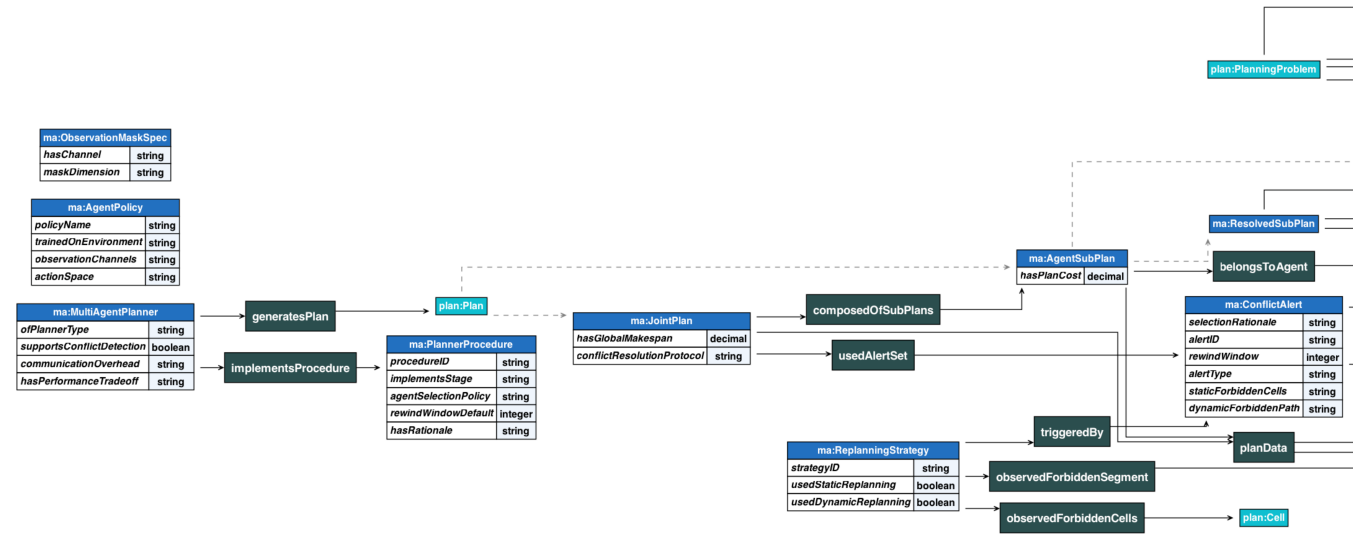
The fundamental unit in a multi-agent system is the agent whose behavior we seek to explain. To model this, we introduce the `ma:Agent` class as a subclass of `plan:ProblemObject`. To formally ground the agent as an entity capable of perception and action, it is also defined as a subclass of `sosa:Platform` from the SOSA ontology (Janowicz et al. 2019). Each agent is defined by its identifier, capabilities, and its initial and goal locations. While simple capabilities can be captured as literals, the `ma:hasCapability` property also formally links to a `cora:Capability` class from the CORA ontology for more structured definitions (Schlenoff et al. 2012).

To represent the state of an agent at a specific moment, the `ma:AgentState` class is created as a subclass to `plan:State`. It captures an agent's location at a point in time using the `ma:agentAt` and `ma:occursAtTime` properties. To align with semantic web standards, all temporal entities, such as the value of `ma:occursAtTime`, are modeled as instances of `time:Instant` from the W3C Time Ontology (Pan and Hobbs 2006). This allows for queries about an agent's status at critical moments, such as the time of a conflict. A key axiom ensures that every agent-specific plan is unambiguously associated with exactly one agent, which is crucial for accountability and explanation: `ma:AgentSubPlan` $\sqsubseteq=$ 1`ma:belongsToAgent.ma:Agent`.

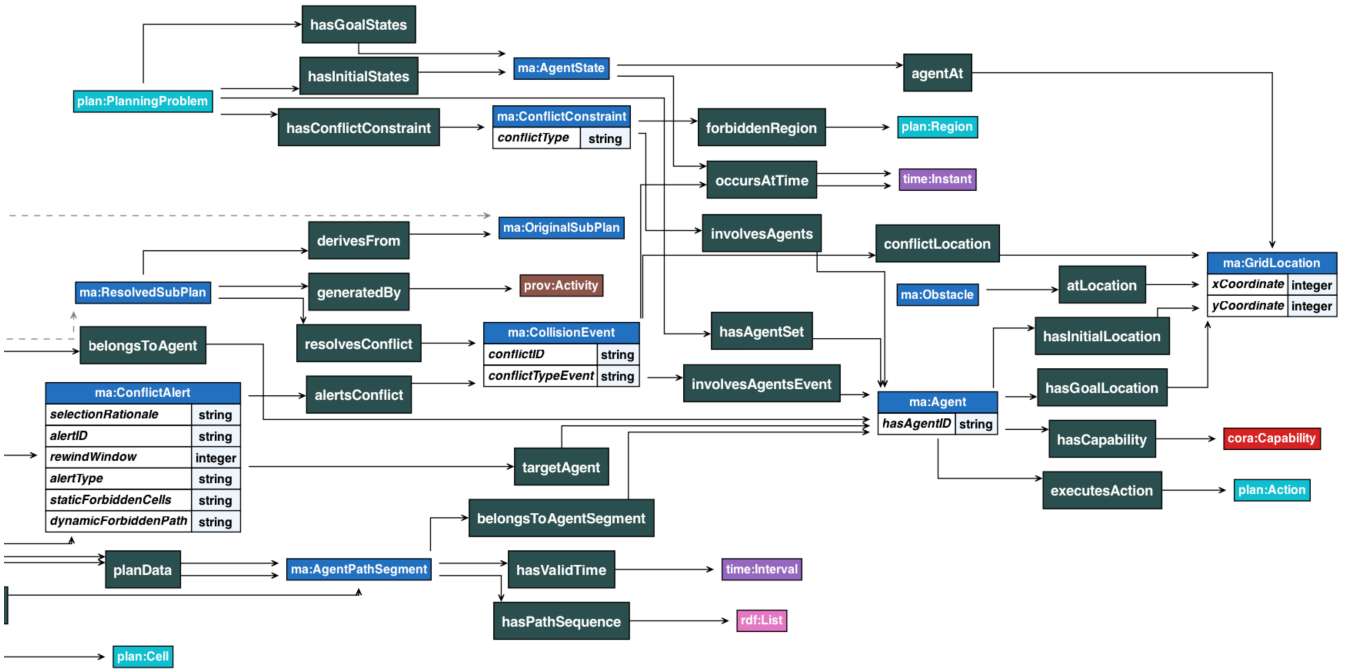
Multi-Agent Plan Representation

In the multi-agent context, a global plan is a composition of individual plans that must be coordinated. Our ontology models this hierarchy with two primary classes derived from `plan:Plan`:

- `ma:AgentSubPlan`: Represents a single agent's plan, which has a `ma:hasPlanCost`. It is further specialized into `ma:OriginalSubPlan` (the initial, conflict-unaware plan) and `ma:ResolvedSubPlan` (a revised plan generated after conflict resolution). This distinction



(a) **Planning and conflict-resolution workflow:** From the initial `plan:PlanningProblem` and `ma:MultiAgentPlanner`, through `ma:PlannerProcedure`, to the generation of an initial `plan:Plan`, conflict detection (`ma:ConflictConstraint`, `ma:CollisionEvent`), alerting (`ma:ConflictAlert`), replanning (`ma:ReplanningStrategy`), and assembly of `ma:JointPlan` and `ma:AgentSubPlan` instances.



(b) **Agent and state modeling:** How `ma:Agent`, `ma:AgentState`, and `ma:GridLocation` capture agent identity, capabilities, and spatial trajectory via `ma:AgentPathSegment`; the incorporation of conflict constraints (`ma:ConflictConstraint`), temporal context (`time:Instant`, `time:Interval`), and provenance (`prov:Activity`) to enable rich, causal explanations.

Figure 1: Complete Multi-Agent Planning Ontology (*maPO*), split into two panels: (a) **Planning and conflict-resolution workflow**, illustrating the end-to-end pipeline from problem definition and procedural plan generation through conflict detection, alerting, strategy selection, and derivation of joint and agent-specific subplans; and (b) **Agent and state modeling**, detailing how agents, their capabilities, and grid-based path segments are represented alongside conflict constraints, temporal instants/intervals, and provenance activities to support on-demand explainability. Dashed arrows denote `rdfs:subClassOf` hierarchies, dark-gray boxes are object properties, and tables list datatype properties (with XSD ranges).

is necessary for explaining *why* a plan changed and is formally captured using the W3C PROV Ontology, as described in the next section.

- `ma:JointPlan`: Represents the final, conflict-free, and globally consistent solution for all agents. It is defined by its constituent subplans via the `ma:composedOfSubPlans` property and its overall efficiency by `ma:hasGlobalMakespan`.

The fine-grained trajectory of each agent is captured by the `ma:AgentPathSegment` class. This class details an agent's location, represented not as a simple string but as an ordered sequence (`rdf:Seq`) of structured grid coordinates (Beckett and McBride 2004). This segment exists over a specific time interval, which is formally represented as a `time:Interval` from the W3C Time Ontology (Pan and Hobbs 2006), defined by a beginning and an end instant. This provides the ground truth for an agent's movement, allowing for the analysis of specific actions like waiting, which occurs when consecutive segments share the same location. The relationship between a plan and its detailed steps is formalized as `ma:AgentSubPlan` $\sqsubseteq$ $\exists$`ma:planData.ma:AgentPathSegment`.

Conflict and Resolution Modeling

The core of our ontology's explanatory power lies in its ability to model the conflict resolution lifecycle. This is achieved through a chain of classes that represent the causal link from problem detection to solution implementation. By reusing the W3C PROV Ontology (Lebo et al. 2013), we make the planner's reasoning process transparent, traceable, and founded on a global standard for provenance.

Detection: A `ma:CollisionEvent` represents a conflict detected by the planner. It captures the essential "what, where, when, and who" of a conflict through properties detailing its time (`ma:occursAtTime`), location (`ma:conflictLocation`), type (`ma:conflictTypeEvent`), and the set of agents involved (`ma:involvesAgentsEvent`).

Alerting: In response, the planner issues a `ma:ConflictAlert`. This class links the abstract problem to a concrete action, specifying which agent is targeted (`ma:targetAgent`) for which specific conflict (`ma:ConflictAlert` $\sqsubseteq$ $\exists$`ma:alertsConflict.ma:CollisionEvent`). It also contains the planner's justification for this choice in the `ma:selectionRationale` property, which is essential for answering competency question C7.

Strategy Selection and Provenance: The alerted agent employs a `ma:ReplanningStrategy`. This action is modeled as a `prov:Activity`, linking it to the alert that prompted it. The strategy generates a new `ma:ResolvedSubPlan`. This new plan is causally linked back to its origin using two critical PROV-O properties: `prov:wasGeneratedBy`, which points to the replanning `prov:Activity`, and `prov:wasDerivedFrom`, which points back to the `ma:OriginalSubPlan` that it replaces.

```
{
  "environment": {
    "gridSize": [R, C],
    "obstacles": [ { "id": obs_id, "
        cell": [r, c] },     ]
  },
  "agents": [
    { "id": agent_id,
      "initialState": { "time": t0, "
          cell": [rs, cs] },
      "goalState":    { "cell": [rg,
          cg] }
    },
  ],
  "agentPaths": [
    { "agent": agent_id,
      "planCost": cost,
      "steps": [ { "time": t, "cell":
          [r, c] },     ]
    },
  ],
  "collisionEvents": [
    { "id": coll_id, "time": t, "type
        ": type,
      "location": [r, c], "agents": [
          ai, aj]
    },
  ],
  "jointPlan": {
    "subplans": [ plan_id1,     ],
    "globalMakespan": T_final
  }
}
```

Listing 1: Minimal overview of the unified MAPF JSON log schema.

4 Explanation Framework and Functionality

To render MAPF planners transparent and trustworthy, we ground all explanations in the *maPO*, utilizing SPARQL as the query language. Any planner that produces the structured JSON trace, as shown in Listing 1, can be ingested without code changes. A lightweight Python script asserts the corresponding RDF triples, and a generic SPARQL endpoint permits runtime querying over this unified graph. The results from these SPARQL queries are then systematically populated into a set of predefined natural language (NL) templates to generate the final human-readable explanations (examples are presented in the Appendix section 4).

Conflict-Centric Analysis (C1, C2) To diagnose conflicts, the query in Listing 2[1] enumerates every collision event involving a target agent. The query retrieves the event's timestamp, type, and involved agents, while also aggregating all the coordinate locations associated with

[1]Prefix anonymized; PURL and public resources will be provided upon acceptance.

Table 2: User Study Results (N=25). The preference for Format B was statistically significant in all tasks (Binomial Test, $p < .001$).

Scenario	Task Question	Preference for Generated Exp. (%)	Clarity of Generated Exp. (Mean $\pm$ SD)
1: RL (2 Agents)	What is Agent 1's final plan?	92.0	4.48 ± 0.82
	How was the conflict resolved?	88.0	4.52 ± 0.77
2: CBS (3 Agents)	What is the global plan summary?	84.0	4.12 ± 1.20
	How and why did Agent 1's plan change?	95.8	4.29 ± 1.06
3: ICBS (7 Agents)	Why did Agent 5 take an inefficient path?	92.0	4.56 ± 0.70
	Why did Agent 3 have a long wait?	100	4.64 ± 0.64

```
PREFIX ma: <http://example.org/ma#>
PREFIX time: <http://www.w3.org/2006/time#>
SELECT ?conflict ?timestamp ?type
    (GROUP_CONCAT(DISTINCT ?otherAgentName; separator=", ") as ?involvedAgents)
    (GROUP_CONCAT(CONCAT("(", STR(?y), ",", STR(?x), ")"); separator=";") as ?locations)
WHERE {
    ?conflict a ma:CollisionEvent ;
        ma:involvesAgentsEvent ma:agent-1, ?otherAgent ;
        ma:occursAtTime/time:inXSDDateTimeStamp ?timestamp ;
        ma:conflictTypeEvent ?type ;
        ma:conflictLocation ?locNode .
    ?locNode ma:xCoordinate ?x ; ma:yCoordinate ?y .
    FILTER(?otherAgent != ma:agent-1)
    BIND(STRAFTER(STR(?otherAgent), "#") AS ?otherAgentName) }
GROUP BY ?conflict ?timestamp ?type
ORDER BY ?timestamp
```

Listing 2: Query for All Conflicts Involving a Specific Agent (C1, C2).

the conflict. By extracting time, type, location, and co-participants, this query facilitates causal investigations.

Causal and Contrastive Explanations (C3, C4) Contrastive and delay-focused queries reveal both what changed and why. The query shown in the Appendix (Listing 6) directly compares an agent's trajectory before and after replanning. To explain why an agent was delayed (C4), the query in Listing 3 identifies wait states in the final plan and connects them back to the specific conflict they were designed to resolve, revealing the other agent involved and the location of the conflict.

Global & Agent-Specific Performance (C6, C8) Assessing overall efficiency and individual contributions is achieved with simple queries that retrieve summative properties from the final plan. These SPARQL queries, detailed in the Appendix (Listings 4 and 5), provide insights into makespan and sum-of-costs.

Tracing the Full Resolution History (C7) For a holistic view, we can reconstruct each replanning event by tracing the provenance of an agent's sub-plans. The query, detailed in the Appendix (Listing _reflst:history), starts with the `OriginalSubPlan` and recursively joins all subsequent `ResolvedSubPlan` instances. For each resolved plan, it follows the `prov:wasGeneratedBy` property to find the replanning activity and rationale, extracting the complete sequence of events and revealing the planner's reasoning (`ma:selectionRationale`) at each step.

Extensibility and Future Applications Our framework's true asset is the ontology, which turns raw planner logs into a flexible, queryable knowledge base, rather than any particular SPARQL example. By decoupling explanation generation from a planner's internal logic, we create a single, reusable foundation for analysis and future research. This base can support a wide range of extensions: visual plan-segmentation via temporal and spatial queries (Almagor and Lahijanian 2020), automated safety-rule enforcement, conflict-driven performance diagnostics, or advanced "what-if" analyses. We believe the ontology serves as a key building block for future work in trustworthy and explainable multi-agent systems.

5 User Study

To evaluate the effectiveness and clarity of the explanations generated by our ontology-driven framework, we conducted a user study comparing our system's output against raw planner logs.

User Study Design

To demonstrate the planner-agnostic nature of our framework, we conducted a within-subjects user study. The study presented participants with three distinct MAPF scenarios, each generated by a different class of MAPF algorithm, showcasing the breadth of our approach.

```
PREFIX ma: <http://example.org/ma#>
PREFIX time: <http://www.w3.org/2006/time#>
SELECT ?startTS ?endTS ?otherAgentName ?conflictTS ?x ?y
WHERE {
    ?resolvedPlan a ma:ResolvedSubPlan; ma:belongsToAgent ma:agent-1 .
    FILTER EXISTS { ma:FinalJointPlan ma:composedOfSubPlans ?resolvedPlan . }
    ?resolvedPlan ma:planData ?seg1, ?seg2 .
    ?seg1 ma:hasPathSequence/rdf:rest*/rdf:first ?locNode .
    ?seg1 ma:hasValidTime/time:hasBeginning/time:inXSDDateTimeStamp ?startTS .
    ?seg2 ma:hasPathSequence/rdf:rest*/rdf:first ?locNode .
    ?seg2 ma:hasValidTime/time:hasBeginning/time:inXSDDateTimeStamp ?endTS .
    ?resolvedPlan ma:resolvesConflict ?conflict .
    ?conflict ma:involvesAgentsEvent ?otherAgent ;
          ma:occursAtTime/time:inXSDDateTimeStamp ?conflictTS ;
          ma:conflictLocation ?locNode .
    ?locNode ma:xCoordinate ?x ; ma:yCoordinate ?y .
    FILTER(?otherAgent != ma:agent-1)
    BIND(STRAFTER(STR(?otherAgent), "#") AS ?otherAgentName) }
```

Listing 3: Explain Why an Agent Waited at a Location (C4).

Scenario 1 (RL-based): A two-agent scenario planned by a reinforcement learning agent, representing modern, learning-based decentralized approaches. **Scenario 2 (CBS):** A three-agent scenario solved by a classic CBS algorithm, a complete and optimal centralized search method. **Scenario 3 (ICBS):** A complex, seven-agent scenario in a congested environment, solved by ICBS to represent state-of-the-art heuristic search planners.

For each task within these scenarios, participants were presented with two explanation formats: **Format A (Raw Data)**, which showed relevant excerpts from a typical planner log (e.g., lists of coordinates, multiple plan versions), and **Format B (Generated Explanation)**, which showed the natural-language output from our system. Participants were then asked to (1) choose which format was clearer for answering the task's question and (2) rate the clarity of Format B on a 5-point Likert scale (1 = Very Unclear, 5 = Very Clear). The study was completed by 25 participants.

User Study Results

The results, summarized in Table 2, show a clear and statistically significant preference for the generated explanations across all scenarios and question types. Across all six tasks, participants chose the generated explanation (Format B) as the clearer format in 140 out of 149 total responses (94.0%). This preference for Format B was statistically significant for every task (Binomial Test, $p < .001$). A binomial test (Wagner-Menghin 2005) validates that this preference is statistically significant and not the result of a random choice. Furthermore, the clarity of the generated explanations was consistently rated very high, with a combined mean score of 4.44 ($SD = 0.88$) out of 5 across all tasks. One-sample Wilcoxon signed-rank (Woolson 2007) tests confirmed that the median clarity ratings for all six tasks were higher than the neutral midpoint of 3 (all $p < .001$). This non-parametric test is ideal for Likert scale data, and our results confirm that the high clarity ratings represent a significant positive sentiment, rather than a neutral or random one.

6 Discussion & Conclusion

In this paper, we introduce the *maPO*, a formal knowledge framework designed to make complex MAPF planners transparent and auditable. Our primary contribution is a unified, planner-agnostic approach that transforms low-level execution logs into a rich, queryable knowledge graph. Building upon this foundation, we developed an explanation framework that uses SPARQL queries to translate the structured data into natural-language summaries. The effectiveness of the explanations produced by this framework was validated through a comprehensive user study, which provided strong statistical evidence that our approach makes raw planner data significantly more understandable. Participants found the generated explanations clearer than raw data, particularly for tasks that would otherwise impose a high cognitive load, such as calculating global plan metrics or parsing multiple plan revisions.

By modeling the explicit causal chain from conflict detection to resolution using established standards like the PROV Ontology, our framework delivers concise, causal explanations for complex agent behaviors, such as waiting or taking detours. This capability confirms that our approach not only works but also scales effectively with environment complexity, making planner decisions transparent. We acknowledge that while our template-based explanations proved effective, future work could incorporate more advanced Natural Language Generation (NLG) techniques for richer, more dynamic narratives. Moreover, the extensibility of our framework opens avenues in real-world robotics: by defining agents as `sosa:Platform` instances, we prepare the ontology to connect with physical robots. The next logical step is to incorporate live data from robot sensors, which would allow the system to explain why an agent reacts to unexpected events, such as a sudden obstacle, or debug communication latencies in coordinated tasks. This creates a full explanation system, from the initial plan to real-time actions, that is essential for building trust and safely deploying autonomous agents in the real world.

References

Almagor, S.; and Lahijanian, M. 2020. Explainable multi agent path finding. In *AAMAS*.

Beckett, D.; and McBride, B. 2004. RDF/XML syntax specification (revised). *W3C recommendation*, 10(2.3).

Bogatarkan, A. 2021. Flexible and explainable solutions for multi-agent path finding problems. *arXiv preprint arXiv:2109.08299*.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Betzalel, O.; Tolpin, D.; and Shimony, E. 2015. Icbs: The improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 6, 223–225.

Brandao, M.; Mansouri, M.; Mohammed, A.; Luff, P.; and Coles, A. 2022. Explainability in multi-agent path/motion planning: User-study-driven taxonomy and requirements. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Damani, M.; Luo, Z.; Wenzel, E.; and Sartoretti, G. 2021. PRIMAL _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2): 2666–2673.

Janowicz, K.; Haller, A.; Cox, S. J.; Le Phuoc, D.; and Lefrançois, M. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56: 1–10.

Kottinger, J.; Almagor, S.; and Lahijanian, M. 2022. Conflict-based search for explainable multi-agent path finding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 692–700.

Lebo, T.; Sahoo, S.; McGuinness, D.; Belhajjame, K.; Cheney, J.; Corsar, D.; Garijo, D.; Soiland-Reyes, S.; Zednik, S.; and Zhao, J. 2013. Prov-o: The prov ontology.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10256–10265.

Muppasani, B.; Gupta, N.; Pallagani, V.; Srivastava, B.; Mutharaju, R.; Huhns, M. N.; and Narayanan, V. 2024. Building a Plan Ontology to Represent and Exploit Planning Knowledge and Its Applications. In *Eighth International Conference on Data Science and Management of Data (CODS-COMAD'24), India*.

Pan, F.; and Hobbs, J. R. 2006. Time ontology in owl. *W3C working draft, W3C*, 1(1): 1.

Ren, J.; Eric, E.; Kumar, T. K. S.; Koenig, S.; and Ayanian, N. 2025. Empirical Hardness in Multi-Agent Pathfinding: Research Challenges and Opportunities. In *Blue Sky paper at 24th International Conference on Autonomous Agents and Multiagent Systems*.

Sartoretti, G.; et al. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters*, 4(3): 2559–2566.

Schlenoff, C.; Prestes, E.; Madhavan, R.; Goncalves, P.; Li, H.; Balakirsky, S.; Kramer, T.; and Miguelanez, E. 2012. An IEEE standard ontology for robotics and automation. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 1337–1342. IEEE.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The Increasing Cost Tree Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 195(C): 470–495.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, 151–158.

Wagner-Menghin, M. M. 2005. Binomial test. *Encyclopedia of statistics in behavioral science*.

Wang, Y.; Duhan, T.; Li, J.; and Sartoretti, G. A. 2025. LNS2+RL: Combining Multi-agent Reinforcement Learning with Large Neighborhood Search in Multi-agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wang, Y.; Xiang, B.; Huang, S.; and Sartoretti, G. 2023a. Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9301–9308. IEEE.

Wang, Y.; Xiang, B.; Huang, S.; and Sartoretti, G. 2023b. SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9301–9308.

Woolson, R. F. 2007. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, 1–3.

Reproducibility Checklist

Instructions for Authors:
This document outlines key aspects for assessing reproducibility. Please provide your input by editing this `.tex` file directly.

For each question (that applies), replace the "Type your response here" text with your answer.

Example: If a question appears as

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
Type your response here
```

you would change it to:

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
yes
```

Please make sure to:

- Replace ONLY the "Type your response here" text and nothing else.
- Use one of the options listed for that question (e.g., **yes**, **no**, **partial**, or **NA**).
- **Not** modify any other part of the `\question` command or any other lines in this document.

You can `\input` this .tex file right before `\end{document}` of your main file or compile it as a stand-alone document. Check the instructions on your conference's website to see if you will be asked to provide this checklist with your paper or separately.

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) yes

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) no

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) Type your response here

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) Type your response here

2.4. Proofs of all novel claims are included (yes/partial/no) Type your response here

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) Type your response here

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) Type your response here

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) Type your response here

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) Type your response here

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) no

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) Type your response here

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) Type your response here

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) Type your response here

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) Type your response here

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) Type your response here

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) Type your response here

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) no

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) Type your response here

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) Type your response here

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) Type your response here

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) Type your response here

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) Type your response here

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) Type your response here

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) Type your response here

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) Type your response here

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) Type your response here

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) Type your response here

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) Type your response here

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) Type your response here

Appendix

SPARQL Query Listings

This appendix contains supplemental SPARQL queries referenced in Section 4.

```
PREFIX ma: <http://example.org/ma#>
SELECT ?makespan WHERE {
    ma:FinalJointPlan ma:
        hasGlobalMakespan ?makespan.
        }
```

Listing 4: Query for Global Makespan (C8).

```
PREFIX ma: <http://example.org/ma#>
SELECT (STRAFTER(STR(?agent), "#") AS
     ?agentName) ?cost WHERE {
    ma:FinalJointPlan ma:
        composedOfSubPlans ?plan.
    ?plan ma:belongsToAgent ?agent ;
          ma:hasPlanCost ?cost. }
```

Listing 5: Query for Per-Agent Final Cost (C6).

Statistical Methods

This appendix details the formulas used to calculate the statistical values presented in the user study results.

Preference Percentage This is the percentage of participants who preferred the generated explanation (Format B) for a given task.

$$\text{Preference } \% = \frac{\text{Number of votes for Format B}}{\text{Total number of votes}} \times 100 \quad (1)$$

Mean and Standard Deviation The mean ($\bar{x}$) and sample standard deviation (s) were calculated for the Likert scale clarity ratings for each task.

$$\text{Mean } (\bar{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (2)$$

$$\text{Standard Deviation } (s) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (3)$$

Where x_i is an individual clarity rating and n is the number of participants who rated that task.

Binomial Test The binomial test was used to determine if the preference for one format was statistically significant. The null hypothesis is that the probability of choosing either format is equal ($p = 0.5$). The probability of observing k successes (e.g., choices for Format B) in n trials is given by the probability mass function:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (4)$$

The reported p-value is the probability of observing a result at least as extreme as the one measured.

One-Sample Wilcoxon Signed-Rank Test This non-parametric test was used to determine if the median of the clarity ratings was significantly different from the neutral midpoint of the Likert scale ($\mu_0 = 3$). The test involves ranking the absolute differences between each rating (x_i) and the hypothesized median (μ_0), and then summing the ranks based on the sign of the difference. The resulting test statistic, W, is compared to a critical value to determine the p-value. This test is suitable for ordinal data and does not assume a normal distribution.

```
PREFIX ma: <http://example.org/ma#>
PREFIX time: <http://www.w3.org/2006/time#>
SELECT ?phase ?timestamp ?x ?y
WHERE {
    BIND(ma:agent-1 AS ?agent)
    {
        ?orig a ma:OriginalSubPlan ;
              ma:belongsToAgent ?agent .
        ?orig ma:planData/ma:hasValidTime/time:hasBeginning/time:inXSDDateTimeStamp ?timestamp ;
              ma:planData/ma:hasPathSequence/rdf:rest*/rdf:first ?cellNode .
        ?cellNode ma:xCoordinate ?x ; ma:yCoordinate ?y .
        BIND("Original" AS ?phase)
    } UNION {
        ?res a ma:ResolvedSubPlan ;
             ma:belongsToAgent ?agent .
        FILTER EXISTS { ma:FinalJointPlan ma:composedOfSubPlans ?res . }
        ?res ma:planData/ma:hasValidTime/time:hasBeginning/time:inXSDDateTimeStamp ?timestamp ;
             ma:planData/ma:hasPathSequence/rdf:rest*/rdf:first ?cellNode .
        ?cellNode ma:xCoordinate ?x ; ma:yCoordinate ?y .
        BIND("Resolved" AS ?phase)
    }
} ORDER BY ?phase ?timestamp
```

Listing 6: Original vs. Resolved Plan Comparison (C3).

```
PREFIX ma: <http://example.org/ma#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX prov: <http://www.w.org/ns/prov#>
SELECT ?planType ?planCost ?timestamp ?x ?y ?rationale
    (GROUP_CONCAT(DISTINCT ?otherAgentName; separator=", ") AS ?otherAgents)
WHERE {
    BIND(ma:agent-1 AS ?agent)
    {
        ?plan a ma:OriginalSubPlan ;
              ma:belongsToAgent ?agent ;
              ma:hasPlanCost ?planCost .
        BIND("Original Plan" as ?planType)
        BIND("N/A" as ?timestamp) BIND("N/A" as ?x) BIND("N/A" as ?y)
        BIND("Initial plan generation" as ?rationale)
    } UNION {
        ?plan a ma:ResolvedSubPlan ;
              ma:belongsToAgent ?agent ;
              ma:hasPlanCost ?planCost ;
              ma:resolvesConflict ?conflict ;
              ma:generatedBy ?activity .
        ?activity prov:used/ma:triggeredBy/ma:selectionRationale ?rationale .
        ?conflict ma:occursAtTime/time:inXSDDateTimeStamp ?timestamp ;
                  ma:conflictLocation ?locNode ;
                  ma:involvesAgentsEvent ?otherAgent .
        ?locNode ma:xCoordinate ?x ; ma:yCoordinate ?y .
        FILTER(?otherAgent != ?agent)
        BIND("Resolved Plan" as ?planType)
        BIND(STRAFTER(STR(?otherAgent), "#") as ?otherAgentName)
    }
}
GROUP BY ?plan ?planType ?planCost ?timestamp ?x ?y ?rationale
ORDER BY ?timestamp
```

Listing 7: Reconstruct an Agent's Full Replanning History (C7).

Natural Language Explanation Templates

Agent's Final Plan

*The final plan for **[Agent Name]** has a cost of **[Cost]**.*
***Summary:** The agent starts at [Start], then [Summary of movements...].*
***Full Path:** [Path string]*

Agent's Conflicts

***[Agent Name]** was involved in these conflicts (highlighted on grid):*

Time	Type	Location	With
...	...	...	...

Conflict Resolution and Plan Change

*A conflict occurred at location **[Locations]**. To resolve the conflict with **[Other Agent Name]**, **[Replanning Agent Name]** changed its plan. This changed the plan cost from **[Original Cost]** to **[New Cost]**.*
***Rationale:** Using a **[Alert Type]** strategy, [Rationale].*

Agent Wait Explanation

***[Agent Name]** had to wait for a total of **[Total Wait]** second(s) in its final plan. This was necessary to avoid a planned conflict with **[Other Agent Name]** that would have occurred at ([X],[Y]) at T=[Conflict Time].*

Global Plan Summary

Here is a summary of the final joint plan:
*The overall **makespan** is **[Makespan]**.*
*The **sum of individual plan costs** is **[Sum of Costs]**.*
*To achieve this solution, a total of **[Replan Count] replans** were required.*