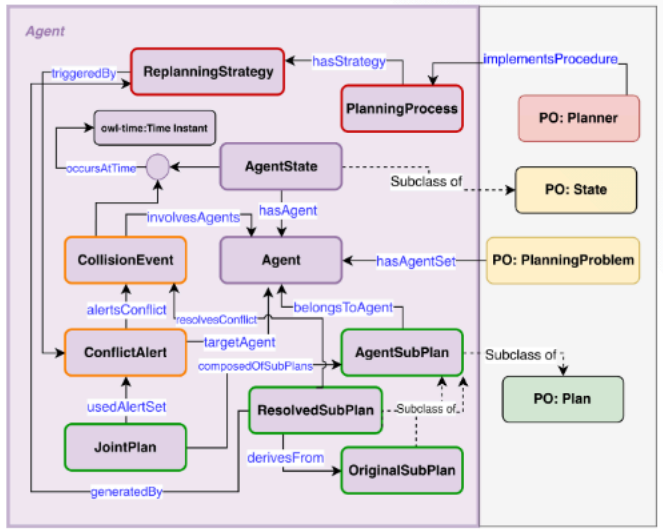
maPO: An Ontology for Multi-Agent Path Finding and Its Usage for Explaining Planner Behaviour

Bharath Muppasani · Ritirupa Dey · Biplav Srivastava · Vignesh Narayanan

Artificial Intelligence Institute of South Carolina (AIISC) · University of South Carolina




MAPF Explanations · Planner-agnostic · Ontology · SPARQL-based



REFERENCE LINKS

PURL purl.org/ai4s/ontology/planning/multi-agent

Website ai4society.github.io/ma-planning-ontology/

Funding. This work was supported in part by a J.P. Morgan Chase Faculty Award, AFOSR Award No. FA9550-24-1-0228, and NSF Award Nos. 2337998 and 2454027.

AAAI 2026 SPRING SYMPOSIUM SERIES · MAKE 2026

Machine Learning and Knowledge Engineering for Knowledge-Grounded Semantic Agents

April 7-9, 2026 · Burlingame, California

A planner-agnostic semantic framework for turning MAPF traces into provenance-aware, SPARQL-queryable explanations.

OUTLINE

Talk Roadmap

03-04 **Problem Setting and Literature Gap**
MAPF Requires Explanations and Existing Explanation Methods Are Fragmented.

05 **Solution Overview**
maPO Turns Logs Into Knowledge.

06-07 **Ontology Background and Competency Questions**
Why Ontologies Matter and Competency Questions for maPO.

08-09 **Ontology Structure and Provenance**
Ontology Clusters and How Plans Evolve.

10-12 **Pipeline, Explanation Interface, and Querying**
Method: From JSON to Queryable KG, SPARQL Explanation Interface, and What the Knowledge Graph Can Answer.

13-15 **Evaluation, Schema Scope, and Conclusion**
Evaluation of maPO Explanations, What maPO Explicitly Models, and Contributions and Next Steps.

PROBLEM SETTING

MAPF Requires Explanations

*MAPF computes coordinated multi-robot motion effectively, yet the **causal basis of replanning decisions** often remains opaque to human operators.*

Operational accountability | Conflict-driven replanning | Human oversight

Joint execution in shared space
Multiple agents follow coordinated trajectories toward distinct goals on the same grid.

Conflict resolution revises behavior
Safe execution may introduce waits, detours, or revised subplans that were not present in the original plan.

Operators require traceable reasons
An explanation must identify which conflict occurred, which agent was affected, and why the resulting behavior changed.

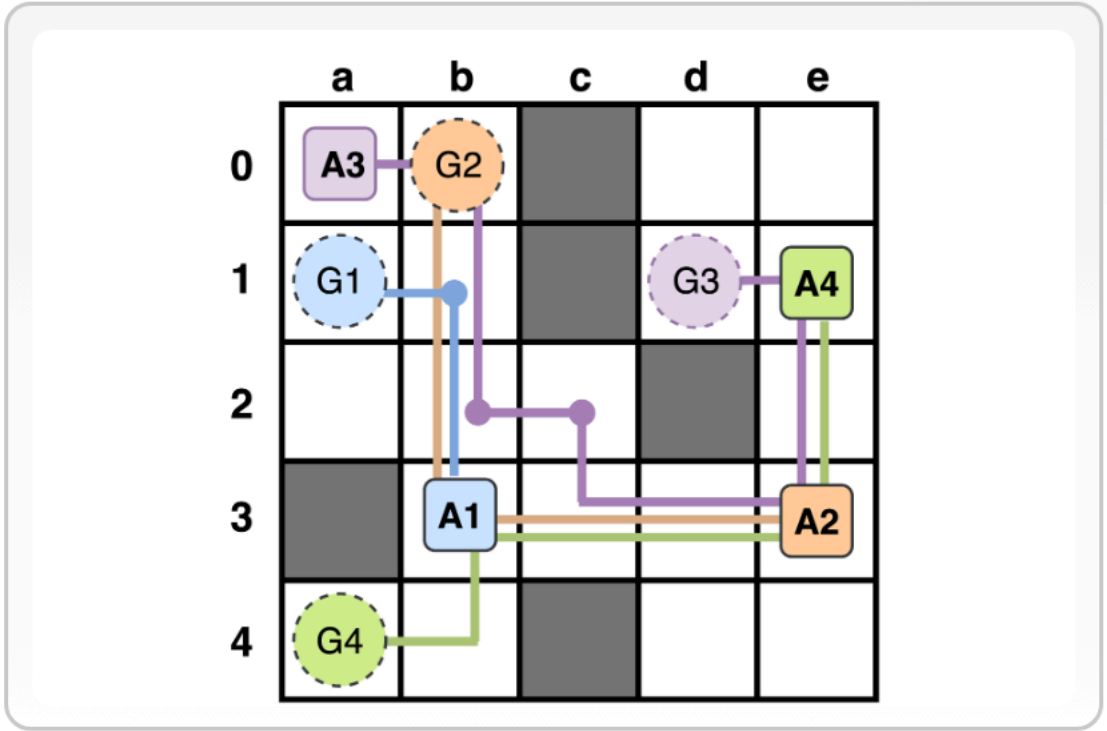


Figure 1. Representative MAPF instance with obstacles, interacting trajectories, and conflict-prone motion on a shared grid.

Conventional planner outputs expose final paths, but they rarely preserve the **semantic rationale for revision** that links a specific conflict to the resolved behavior.

LITERATURE GAP

Existing Explanation Methods Are Fragmented

*Segmentation snapshots, stakeholder taxonomies, and logic-based why queries each explain **one part of the problem**, but MAPF still lacks a unified formal model that links them to planner traces.*

Segmentation-based visual explanations [1, 2]

These help humans verify safety, but they do not preserve provenance, replanning rationale, or reusable semantics.

User-centered explanation taxonomies [3]

These clarify what stakeholders want to know, but not a machine-readable structure that answers those questions across planners.

Logic-based why / why-not queries [4]

These can answer specific replanning questions, yet MAPF still lacks a shared semantic representation spanning traces, conflicts, and revisions.

What is still missing

A **unified semantic framework** that formalizes MAPF entities and supports visual, logical, and contrastive explanations from the same trace.

Planner trace in JSON schema

```
{
  "environment": {
    "gridSize": [R, C],
    "obstacles": [{ "id": obs_id, "cell": [r, c] }]
  },
  "agents": [{
    "id": agent_id,
    "start": { "t": t0, "cell": [rs, cs] },
    "goal":  { "cell": [rg, cg] }
  }],
  "collisions": [{
    "id": coll_id,
    "t": t,
    "type": "vertex" | "edge",
    "loc": [r, c],
    "agents": [ai, aj]
  }],
  "alerts": [{
    "id": alert_id,
    "conflict": coll_id,
    "agent": agent_id,
    "type": alert_type,
    "reason": text
  }],
  "jointPlan": {
    "subplans": [plan_id],
    "makespan": T
  }
}
```

Figure 2. Planner logs contain rich facts, but those facts are not yet organized as a shared explanation model.

SOLUTION

maPO Turns Logs Into Knowledge

*maPO combines a **planner-agnostic ontology schema** with a **SPARQL-based explanation framework**.*
Structured MAPF traces are mapped to RDF triples, then queried without modifying the underlying planner.

Ontology schema

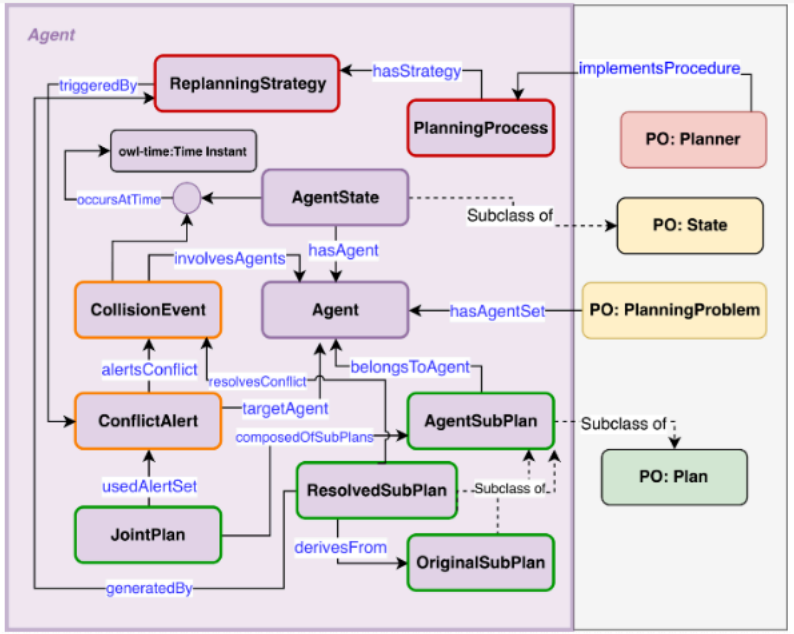


Figure 3a. maPO extends the Planning Ontology with MAPF-specific constructs for agents, conflicts, alerts, strategies, and revised plans.

System architecture

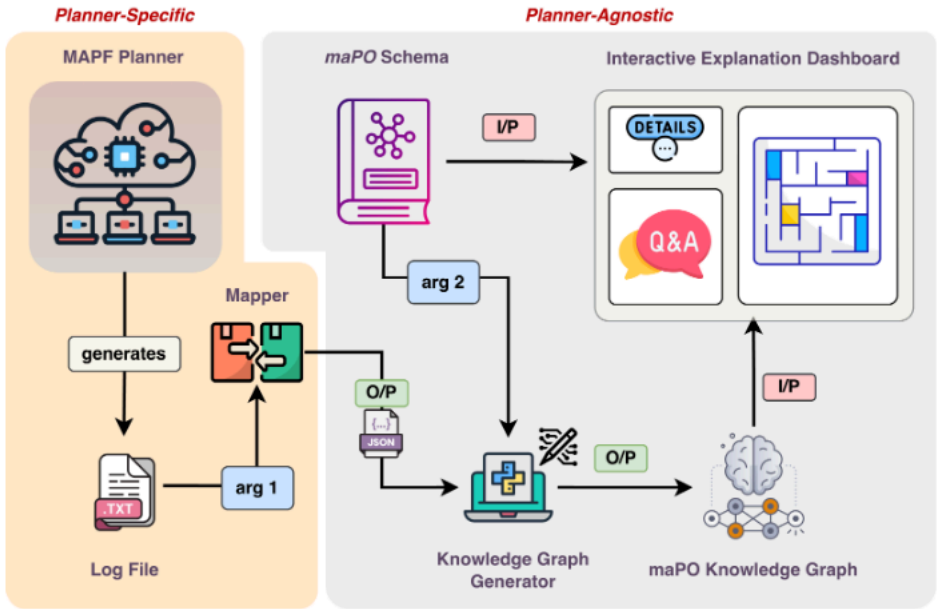


Figure 3b. Structured JSON traces are mapped to RDF and served to a web platform for SPARQL-based explanation generation.

ONTOLOGY BACKGROUND

Why Ontologies Matter

*An ontology specifies the **classes**, **properties**, and **constraints** of a domain before any instance data is added.*

Shared vocabulary

Classes such as `Person`, `Man`, and `Women` define the schema-level vocabulary of the domain.

Reusable relations

Relations such as `subClassOf`, `instanceOf`, and `hasWife` specify how classes and individuals are connected.

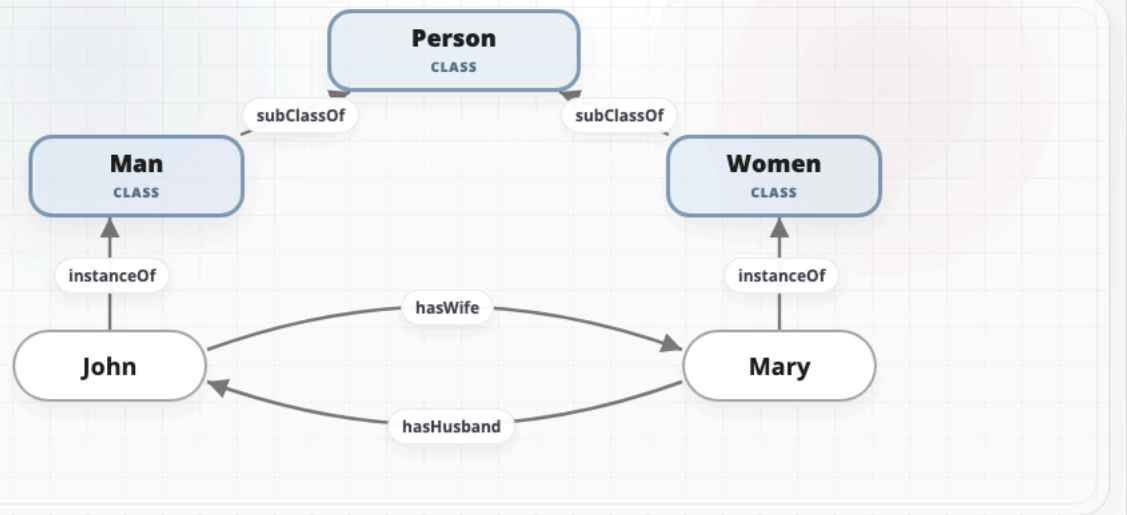
Reasoning and queryability

Because the schema is explicit, instances such as `John` and `Mary` can be interpreted and queried against the same semantics.

Why this helps MAPF

In maPO, the same principle turns an **agent**, a **collision event**, a **replanning strategy**, and a **resolved subplan** into first-class semantic objects rather than disconnected log entries.

EXAMPLE ONTOLOGY SCHEMA



Class hierarchy `Person` is the superclass; `Man` and `Women` are subclasses.

Instance assertions `John` and `Mary` are individuals typed through `instanceOf`.

Object properties `hasWife` and `hasHusband` relate the two individuals.

The same pattern underlies maPO: define classes and relations for the planning domain, then populate execution traces as semantically linked instances.

Competency Questions for maPO

We formulate eight MAPF-specific competency questions to define the explanatory scope of maPO. Together, they specify the entities, relations, and provenance links that the ontology must represent and support through SPARQL.

8 CQs | grouped by explanation task | ontology design requirements

CONFLICT DETECTION AND ALERTS

- C1 Which **CollisionEvents**, including **time, type, location, and involved agents**, were detected during planning?
- C2 For a given **CollisionEvent**, which agent(s) received a **ConflictAlert**?

PLAN REVISION AND CAUSALITY

- 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**?

STRATEGY AND PLANNER RATIONALE

- C5 For a given **ConflictAlert**, which **ReplanningStrategy** did the agent use?
- C7 Why was a particular agent chosen to replan, that is, what was the planner's **selectionRationale**?

COSTS AND FINAL OUTCOME

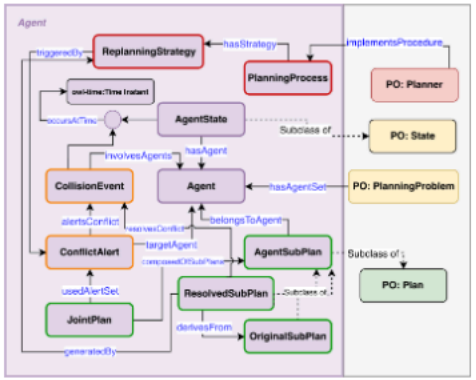
- C6 What was the **cost change** associated with a revised **AgentSubPlan**?
- C8 What is the final **JointPlan** after conflicts are resolved, and what is its overall **makespan**?

These questions determine the **required classes, properties, and causal links** in maPO; because the ontology extends the **Planning Ontology [7]**, it remains aligned with broader planning abstractions already defined there.

Ontology Clusters

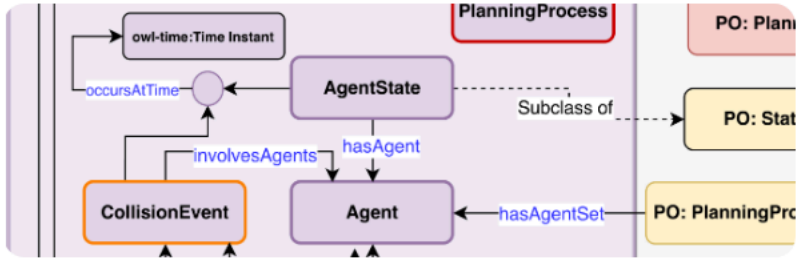
*The full maPO schema is most readable as four coordinated clusters: **agent concepts**, **conflict and resolution**, **plan hierarchy**, and **planner concepts**. Together, these clusters expose the causal structure required for MAPF explanation.*

Canonical ontology map



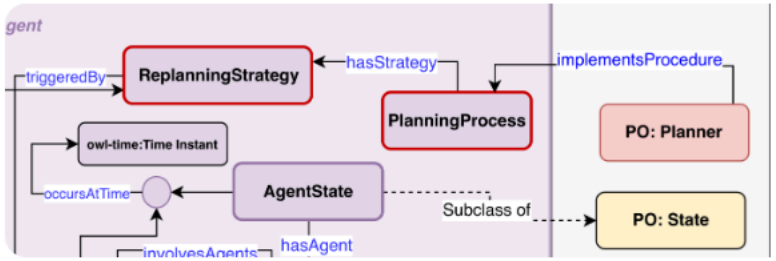
Reading the map

This decomposition shows how maPO links **agent state**, **conflict detection and resolution**, **plan revision**, and **planner abstractions** within one queryable schema.



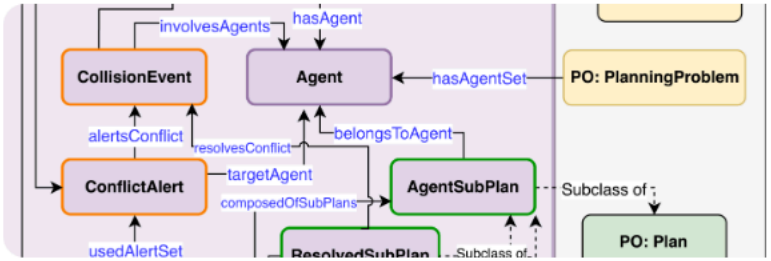
Agent concepts C1 C2 C4

`Agent`, `AgentState`, and temporal links keep each robot grounded in location and time.



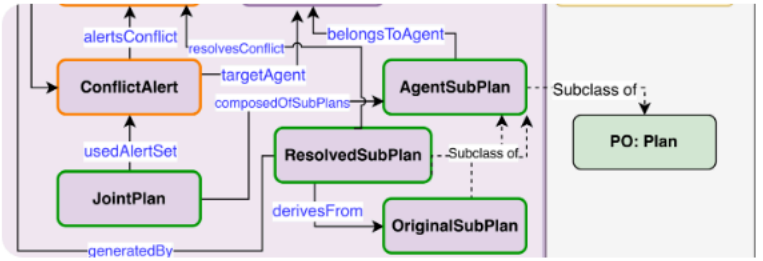
Planner concepts C5 C7

`plan:Planner`, `plan:State`, and `plan:PlanningProblem` link maPO back to the planning stack.



Conflict & resolution C1 C2 C4 C5 C7

`CollisionEvent`, `ConflictAlert`, and `ReplanningStrategy` explain why replanning happens.



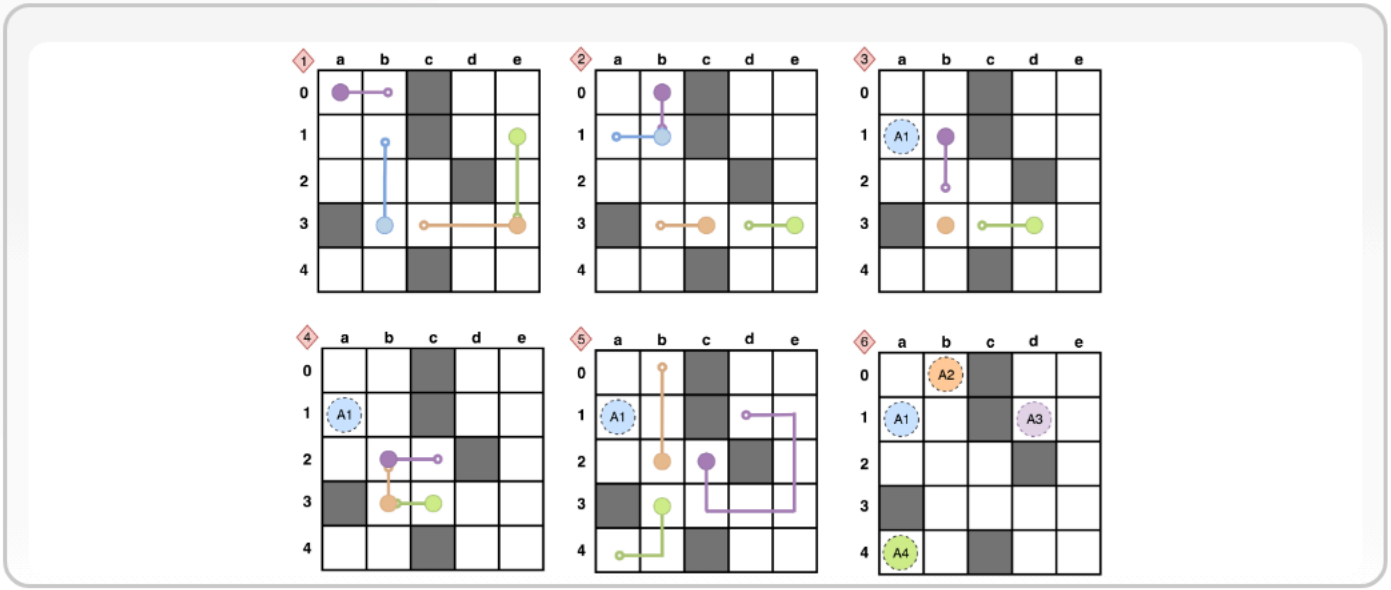
Plan hierarchy C3 C4 C6 C8

`OriginalSubPlan`, `ResolvedSubPlan`, and `JointPlan` preserve before/after structure.

PROVENANCE WALKTHROUGH

How Plans Evolve

*maPO models replanning as a provenance-preserving transformation: each **ResolvedSubPlan** remains linked to the **OriginalSubPlan**, the **CollisionEvent** that invalidated it, and the **ConflictAlert** that initiated repair.*



The segmented storyboard turns one joint execution into a short sequence of collision-free snapshots.

Figure 4. Plan-segmentation based visual explanation: the joint plan is decomposed into sequential, collision-free snapshots that can be inspected for safety.

OriginalSubPlan — The initial route stays preserved after replanning.

↓

CollisionEvent — The ontology records **type, time, location, and agents** for the event that broke the path.

↓

ConflictAlert — The alert identifies **which agent replans** and why.

↓

ResolvedSubPlan — The new route points back to the original path and the conflict it resolves.

This is the missing MAPF audit trail: the graph stores **what changed**, **what caused it**, and **which segment was replaced**.

PIPELINE

Method: From JSON to Queryable KG

MAPF Planner
agents · paths · conflicts

→

JSON Log
canonical execution trace

→

generate_kg.py
Python · RDFLib · triple assertion

→

TTL Knowledge Graph
Turtle / OWL / JSON-LD

→

SPARQL Endpoint
queries · explanation templates

What enters the mapper

Any planner that emits the structured JSON trace can be ingested without code changes. The mapper asserts RDF triples and exposes the same graph through a generic SPARQL endpoint.

Input schema

```
{
  "environment": {
    "gridSize": [R, C],
    "obstacles": [{ "id": obs_id, "cell": [r, c] }]
  },
  "agents": [{
    "id": agent_id,
    "start": { "t": t0, "cell": [rs, cs] },
    "goal":  { "cell": [rg, cg] }
  }],
  "collisions": [{
    "id": coll_id,
    "t": t,
    "type": "vertex" | "edge",
    "loc": [r, c],
    "agents": [ai, aj]
  }],
  "alerts": [{
    "id": alert_id,
    "conflict": coll_id,
    "agent": agent_id,
    "type": alert_type,
    "reason": text
  }],
  "jointPlan": {
    "subplans": [plan_id],
    "makespan": T
  }
}
```

Figure 5a. JSON trace excerpt consumed by the KG generator.

System architecture

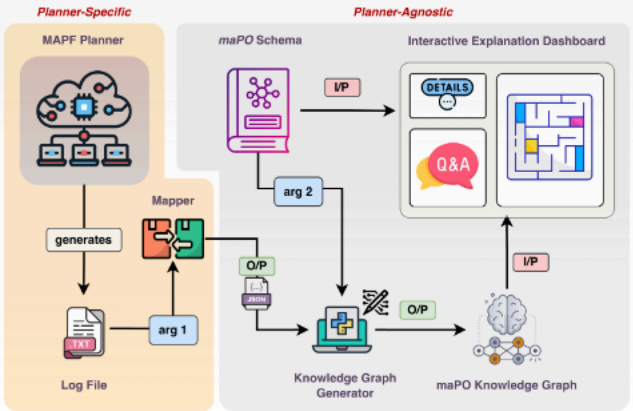


Figure 5b. Automated flow from planner logs to ontology-backed explanations.

READ THE FLOW IN 3 MOVES

Planner-specific input
We begin with raw MAPF traces and optional planner arguments, not a custom explanation API.

Ontology mapping
`generate_kg.py` binds those fields to ontology relations and asserts RDF triples automatically.

Planner-agnostic output
The same knowledge graph then drives SPARQL queries, replay views, and operator-facing explanations.

```
python generate_kg.py --log_file <json> --ontology <ttl> --output <ttl>
```

SPARQL results are then populated into **natural-language explanation templates**, producing **human-readable answers** from the same graph.

EXPLANATION INTERFACE

SPARQL Explanation Interface

*The OMEGA interface **[8]** turns SPARQL results over maPO into visual and natural-language explanations grounded in the same knowledge graph.*

Load a maPO RDF graph

The platform opens the generated knowledge graph and reconstructs the run from ontology instances rather than planner-specific viewers.

Inspect collision, alert, and plan context

The operator can inspect the grid, the conflict location, the affected agents, and the revised subplan from one graph-backed view.

From SPARQL to explanation text

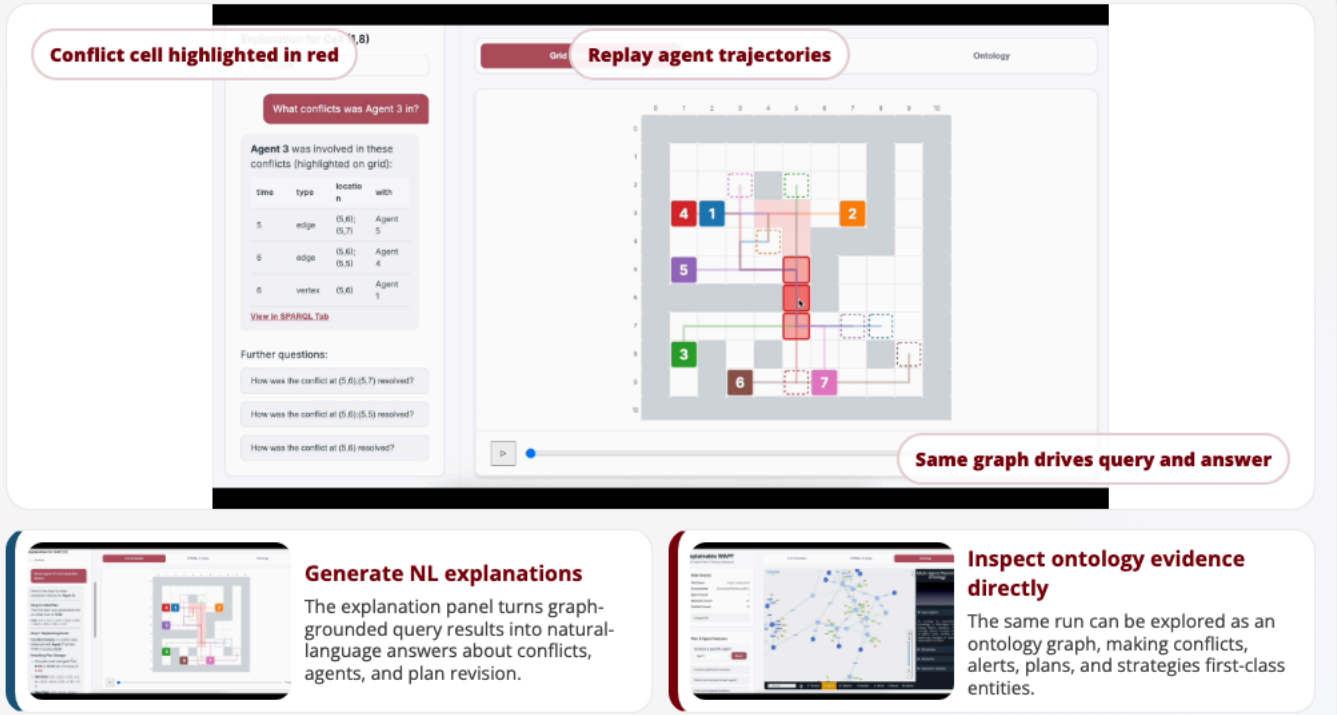
Query results are rendered as explanation templates, so each answer stays grounded in explicit ontology entities and relations.

Q: Why did Agent-3 reroute?

A: The revised subplan resolves a recorded conflict and points back to the alert, strategy, and original plan through provenance links.

DEMO VIDEO SNAPSHOTS

Watch demo



Generate NL explanations

The explanation panel turns graph-grounded query results into natural-language answers about conflicts, agents, and plan revision.

Inspect ontology evidence directly

The same run can be explored as an ontology graph, making conflicts, alerts, plans, and strategies first-class entities.

Figure 6. Demo snapshots show replay, conflict inspection, and ontology evidence from the same maPO graph.

QUERY EXAMPLE

What the Knowledge Graph Can Answer

Concrete SPARQL query

```
SELECT ?type ?time ?x ?y WHERE {
  ?rsp a ma:ResolvedSubPlan ;
       ma:belongsToAgent ma:Robot-1 ;
       ma:resolvesConflict ?c .
  ?c ma:conflictTypeEvent ?type ;
     ma:occursAtTime/time:inXSDDateTimeStamp ?time ;
     ma:conflictLocation ?loc .
  ?loc ma:xCoordinate ?x ;
       ma:yCoordinate ?y .
}
```

Which CollisionEvents were detected? **Who received the ConflictAlert?**

What is the final JointPlan?

AAAI-26 OMEGA LINKS

Paper: https://doi.org/10.1609/aaai.v40i48.42367

Demo video: https://shorturl.at/298j5

The same graph supports conflict-centric, causal, contrastive, global, and provenance queries.

01
C1-C2 Conflict-centric analysis
Which CollisionEvents occurred, and which agent received the ConflictAlert?

02
C3-C4 Causal and contrastive
What changed between the original and final plan, and why did an agent wait or reroute?

03
C5-C7 Strategy and provenance
Which strategy was used, and why was that agent selected to replan?

04
C6-C8 Global summaries
What cost changed, and what is the final JointPlan and makespan?

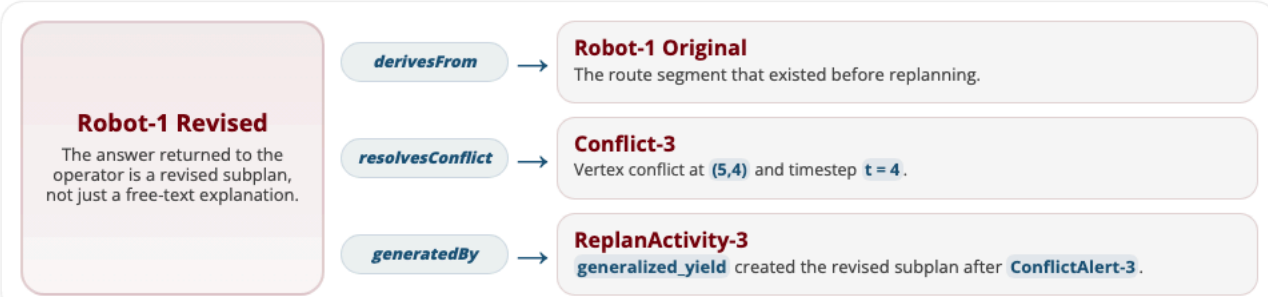


Figure 7. One SPARQL query can be rendered as a readable provenance chain for an operator.

The ontology supports **any query written in SPARQL**. **OMEGA** provides a curated set of **pre-canned queries** for interactive use, but the graph itself is not limited to those templates.

EVALUATION

Evaluation of maPO Explanations

We evaluate maPO explanations in two complementary ways: automated readability analysis for cognitive load and a within-subjects user study against raw planner logs.

Readability evaluation setup

18 generated explanations from a 20-agent RL scenario on an **11x11** grid with **55 obstacles**, covering the full range of competency questions **C1-C8**.

FRE

94.39

very easy to read

Higher is better for readability.

ARI

4.87

early grade level

Lower means less linguistic burden.

CLI

1.13

low complexity

Also indicates easy comprehension.

User study setup

28 participants compared **Format A: raw planner logs** with **Format B: maPO-generated explanations** across three scenarios drawn from **search-based planners such as CBS [5]** and **learning-based** planners.

RL-based · CBS · ICBS · 6 tasks

95.2%

preferred maPO explanations

159 of 167 recorded preferences favored Format B over raw logs.

4.40 / 5

mean clarity rating

Overall clarity score with `SD = 0.90`.

All six tasks were statistically significant: Binomial tests showed preference for Format B with $p < .001$, and Wilcoxon signed-rank tests showed clarity ratings above neutral for every task with $p < .001$.

The generated explanations impose **low linguistic burden** even when they describe **conflicts**, **revised plans**, and provenance over complex MAPF traces.

Low cognitive load · **high user preference** · **evidence beyond a single planner**

SCHEMA SCOPE

What maPO Explicitly Models

maPO extends the Planning Ontology with MAPF-specific entities needed for explanation: agents, subplans, conflicts, alerts, strategies, provenance, and temporal grounding.

Agent

MA:AGENT + MA:AGENTSTATE

identifier, capability, initial and goal location

SubPlan

ORIGINAL / RESOLVED / JOINT

path segments, plan cost, and overall makespan

Conflict

MA:COLLISIONEVENT

type, time, location, and involved agents

Alert

MA:CONFLICTALERT

target agent and planner selectionRationale

PROV-O + TIME

STRATEGY, PROVENANCE, TEMPORALITY

ReplanningStrategy, prov:wasDerivedFrom, prov:wasGeneratedBy, and time:Instant

These constructs make the paper's competency questions answerable over one graph.

CONCLUSION

Contributions and Next Steps

Thank You! Questions?

1 Planner-agnostic maPO schema

A unified ontology for MAPF traces, conflicts, revisions, and provenance across planner families.

SCHEMA

2 Standards-based ontology alignment

Alignment with Planning Ontology, PROV-O, SOSA, and TIME keeps the model interoperable and extensible.

STANDARDS

3 SPARQL-driven explanation framework

The same graph supports segmentation views, provenance walkthroughs, and generated explanation templates.

FRAMEWORK

4 Readability and user-study validation

We evaluate explanation quality through readability analysis and a within-subject user study against raw traces.

EVALUATION

FUTURE TUTORIAL ANNOUNCEMENT

Planning Ontology: A Tutorial on Knowledge Representation and Explainable Planning

ICAPS 2026 | June 27 to July 2 | Dublin, Ireland

NEXT STEPS

- Adaptive natural-language generation grounded in maPO.
- Multimodal and personalized explanations for different users.
- Real-time, sensor-grounded integration with robotic platforms.

Website | Paper | Video

REFERENCES

[1] Almagor, S. and Lahijanian, M. (2020) 'Explainable Multi Agent Path Finding', *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*, pp. 34–42.

[2] Kottinger, J., Almagor, S. and Lahijanian, M. (2022) 'Conflict-Based Search for Explainable Multi-Agent Path Finding', *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1), pp. 692–700.

[3] Brandao, M. et al. (2022) 'Explainability in Multi-Agent Path/Motion Planning: User-study-driven Taxonomy and Requirements', *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS '22)*, pp. 172–180.

[4] Bogatarkan, A. (2021) 'Flexible and Explainable Solutions for Multi-Agent Path Finding Problems', *EPTCS*, 345, pp. 240–247.

[5] Sharon, G. et al. (2015) 'Conflict-Based Search for Optimal Multi-Agent Pathfinding', *Artificial Intelligence*, 219, pp. 40–66.

[6] Boyarski, E. et al. (2015) 'ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding', *Proceedings of the International Symposium on Combinatorial Search*, 6(1), pp. 223–225.

[7] Muppasani, B. et al. (2024) 'Building a Plan Ontology to Represent and Exploit Planning Knowledge and Its Applications', *Proceedings of the Eighth International Conference on Data Science and Management of Data (CODS-COMAD '24)*.

[8] Muppasani, B., Dey, R., Srivastava, B., & Narayanan, V. (2026). 'OMEGA: An Ontology-Driven Tool for Explaining Multi-Agent Path Finding'. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(48), pp. 41643-41645. https://doi.org/10.1609/aaai.v40i48.42367

The slides are designed with the help of Agentic AI tools.