Discover Data

RESEARCH Open Access

Building a planning ontology to represent and exploit planning knowledge and its aplications

Bharath Chandra Muppasani[1*], Nitin Gupta[1], Vishal Pallagani[1], Biplav Srivastava[1], Raghava Mutharaju[2], Michael N. Huhns[1] and Vignesh Narayanan[1]

*Correspondence:
Bharath Chandra Muppasani
bharath@email.sc.edu
[1]University of South Carolina, Columbia, South Carolina, USA
[2]IIIT-Delhi, Delhi, India

Abstract
Ontologies are known for their ability to organize rich metadata, support the identification of novel insights via semantic queries, and promote reuse. In this paper, we build a comprehensive ontology for the problem of *automated planning*, where the objective is to find a sequence of actions that will transform a given initial state of an environment to a desired goal state. We hypothesize that the large number of available planners and diverse planning domains carry valuable information that can be leveraged to enable ontology applications, including planner selection and explanation generation. To this end, we use open data on planning domains and planners to construct the most comprehensive planning ontology to date. This construction is based on supported competency questions that cover different aspects of automated planning, such as planner capabilities, domain requirements, problem structure, plan evaluation, and explanation generation. We then demonstrate its applications in two practical use cases: planner selection and plan explanation. We have also made the ontology and associated resources available to the AI and data communities to promote further research.

Keywords Ontology, Automated planning, Planner selection, Explanation

1 Introduction

Automated planning, a sub-field of Artificial Intelligence (AI), focuses on finding a sequence of actions that transition an agent from an initial state to a desired goal state [1]. The ability to generate plans and make decisions in complex domains, such as robotics, logistics, and manufacturing, has led to significant progress in the automation of planning (refer to workshop series on applications called SPARK [2]). For example, planning algorithms can facilitate tasks ranging from household chores to precision surgeries in robotics or optimize delivery routes in logistics, thereby reducing costs and improving efficiency [3, 4]. The field of automated planning features numerous planners, search algorithms, heuristics, and diverse planning domains. Each planner, combined with specific search algorithms and heuristics, generates plans of varying quality, cost, and optimality. The International Planning Competition (IPC) serves as a benchmark for evaluating planners and their performance across a wide range of domains. This rich empirical data–ranking planners based on their performance–offers valuable insights into identifying tunable parameters to improve planner configurations and applicability to specific domains. Traditionally, planner performance improvements rely on manually exploring and curating combinations of algorithms, heuristics, and configurations to identify optimal setups. While this approach has been effective, it is time-consuming and lacks scalability in a field with an ever-expanding range of planners, heuristics, and planning domains. Moreover, the absence of a structured framework to represent this information limits the ability to reason efficiently about planners and their suitability for specific tasks.

To address this gap, we propose constructing a `Planning Ontology`, the most comprehensive ontology for AI planning to date. Ontologies provide a formal representation of concepts and their relationships, enabling systematic reasoning, knowledge sharing, and reuse [5]. The proposed ontology captures the features of a domain and the capabilities of planners, facilitating tasks such as reasoning with existing planning problems, identifying similarities, suggesting planner configurations, and generating explanations. Planning ontology can also be a useful resource for the creation of new planners as it captures essential information about planning domains and planners, which can be leveraged to design more efficient planning algorithms. Furthermore, ontology can promote knowledge sharing and collaboration within the planning community.

In the field of automated planning, several efforts have been made to develop ontologies that enhance the understanding of planners' capabilities. For instance, Plan-Taxonomy [6] proposed a hierarchical classification aimed at organizing and explaining planner functionalities, while PLANET [7] provided a comprehensive ontology representing plans in real-world domains to support knowledge sharing and application development. However, these prior ontologies offer limited support for reasoning about planner performance or adaptability across diverse planning domains. Moreover, the restricted accessibility and narrow coverage of ontologies such as PLANET, being closed-source, have hindered their reuse and extension by the research community. These limitations motivated our work to design an open, extensible ontology that captures not only the structure of planning models but also the relationships between domains, problems, planners, and plans, enabling semantic reasoning for tasks such as planner selection and plan explanation.

This paper builds on our preliminary work presented at a non-archival workshop [8]. It outlines the methodologies for constructing an ontology to represent *classical* AI planning domains, leveraging information obtained from the IPC and supporting practical usability requirements like explainability (more details in preliminaries Sect. 2.1). Building a planning ontology using data from IPC offers several benefits, such as comprehensive coverage of planning domains, a rich source for various benchmark evaluation metrics, and documentation for planners. However, the ontology is not limited to the Planning Domain Definition Language (PDDL) representation or domains in IPC and can easily be extended to any. In our current work, we extended our preliminary work presented at a non-archival workshop [8].

Our contributions are at the intersection of ontologies and AI planning and can be summarized as follows.

- Building Planning Ontology: To the best of our knowledge, we propose the most comprehensive open-source ontology for AI planning that can be used to represent and organize knowledge related to planning problems. We designed the competency questions to ensure that our ontology provides a structured way to capture the relationships between different planning concepts, enabling more efficient and effective knowledge sharing and reuse.
- Demonstrating Usecase 1: Identifying Most Promising Planner: We demonstrate the ontology's usage for identifying the most promising planner, in terms of past performance, for a specific planning domain using data from IPC.
- Demonstrating Usecase 2: Explanation Generation: We demonstrate the usage of ontology to extract relevant information to generate explanations for the plans generated by automated planners.

In the remainder of the paper, we start with preliminaries about automated planning and IPC. Next, we provide an overview of the existing literature on ontologies for automated planning. Following this, we present a detailed description of the ontology construction process and demonstrate two use cases of the proposed ontology. We conclude with future research directions.

2 Preliminaries

In this section, we describe the necessary background for automated planning and the significance of the International Planning Competition.

2.1 Automated planning

Automated planning or AI planning [1] is formally defined as a tuple (*S*, *A*, *T*, *I*, *G*), where: *S* is the set of possible states of the world, *A* is the set of possible actions that can be taken, *T* is the transition function that describes the effects of taking an action on the current state of the world, *I* is the initial state of the world, *G* is the desired goal state.

Using this notation, the problem of automated planning can be framed as finding a sequence of actions $\prec a_1, a_2, ..., a_k \succ$ that will transform the initial state *I* into the goal state *G*, while respecting any constraints or limitations on the actions. An AI planning problem is often defined in terms of a domain and a problem instance. The domain defines the possible actions that can be taken and the effects of each action, while the problem instance specifies the initial state of the world and the desired goal state. Classical planning is the simplest form of planning where actions have unit cost and take unit time, and all state information are modeled using predicates [1]. Various techniques can be used to solve the planning problem, such as search algorithms, constraint-based reasoning, and optimization methods. These techniques involve exploring the space of possible plans and selecting the one that satisfies the objective and any constraints. Figure 1 illustrates an automated planning scenario for the blocksworld domain, where the initial state can be transformed into the desired goal state by executing the plan with a sequence of actions.

Attributes modeled about a domain.

1. Requirements: A list of requirements that the planner must satisfy to solve the given domain, e.g., *typing* in blocksworld with types.
2. Predicates: Define world properties, e.g., | (on b1 b2)| in blocksworld.

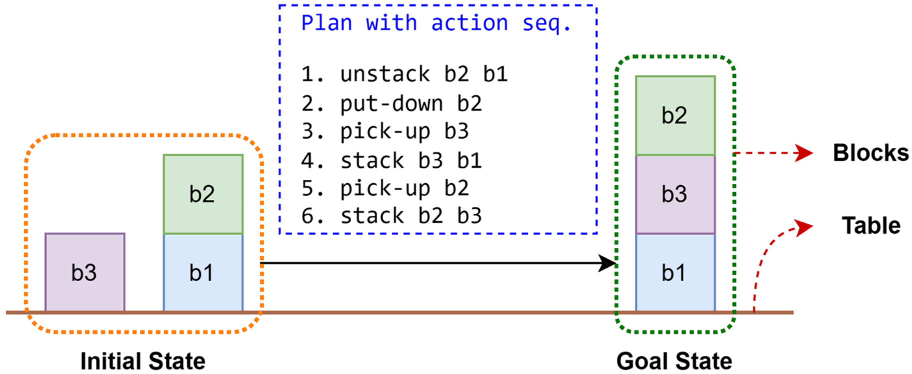

Fig. 1 Demonstration of automated planning problem with blocksworld domain example

3. Actions: Units of change with preconditions and effects, e.g., |unstack b2 b1| in blocksworld.
4. Preconditions: Conditions for action execution, e.g., `(on b1 b2)` for |unstack b2 b1|.
5. Effects: Post-action world changes, e.g., |(not (on b1 b2))| after |unstack b2 b1|.
6. Constants: Fixed values, e.g., *table* in blocksworld.
7. Types: Classifications based on attributes, e.g., |(on ?x - block ?y - block)| in typed blocksworld.

Attributes modeled about a problem instance from a domain.

1. Name: The name of the planning problem.
2. Domain: The name of the planning domain that the problem belongs to.
3. Objects: A list of objects that are present in the planning problem. Objects are typically defined in terms of their type and name. In the example shown in Figure 1, objects are b1, b2, and b3.
4. Initial State: A description of the initial state of the world, including the values of all relevant predicates. Figure 1 represents an example initial state.
5. Goal State: A description of the desired goal state of the world, including the values of all relevant predicates. Figure 1 represents an example goal state.

2.2 International planning competition

The International Planning Competition (IPC) is essential for evaluating planning systems and promoting new methodologies across multiple tracks (e.g., classical, numeric, probabilistic, HTN). Its curated domains and public leaderboards expose strengths and weaknesses of competing planning systems, and the benchmarks, reused across multiple IPC editions, have become a *gold standard* in the planning community for fair and reproducible comparison.

For our experiments, we use the 14 classical domains from IPC-2011: *scanalyzer, elevators, transport, parking, woodworking, floortile, barman, openstacks, nomystery, pegsol, visitall, tidybot, parcprinter*, and *sokoban*. This extensive set demonstrates the competition's breadth and can be further expanded in future work. We chose IPC 2011 for its diverse and extensive set of domains, reflecting a wide range of real-world applications and providing a comprehensive basis for evaluating planning systems. Evaluations on two domains per IPC edition can be found in Appendix Table A1.

Planning Ontology is Data-agnostic Our ontology design is *data-agnostic*: it models planning *concepts and relations* (domains, problems, plans, planner capabilities and performance) independently of any specific IPC year. As a result, the same schema can ingest newer benchmarks and planner results from the latest IPC editions with minimal effort (i.e., by rerunning the existing extraction/mapping scripts on newer result sets).

3 Related work

The use of ontology-based knowledge representation and reasoning has been extensively studied in various domains, including automated planning. This section focuses on the applications of ontology-based knowledge representation and reasoning in the context of planning and related domains. In [9], an ontology is constructed for the Joint Forces Air Component Commander (JFACC) to represent knowledge from the air campaign domain. The ontology is modularized to facilitate data organization and maintenance, but its applicability is domain-specific, unlike our approach. In [10], the authors automate the knowledge discovery workflow using ontology and AI planning, creating a Knowledge Discovery (KD) ontology to represent the KD domain and converting its variables to a Planning Domain Definition Language (PDDL) format to obtain the PDDL domain. The ontology's objects represent initial and goal states, forming the KD task, which represents a specific problem. The authors use the Fast-Forward (FF) planning system to generate the required plans.

In a survey of ontology-based knowledge representation and reasoning in the planning domain, [11] suggests that knowledge reasoning approaches can draw new conclusions in non-deterministic contexts and assist with dynamic planning. In [7], a reusable ontology, PLANET, is proposed for representing plans. PLANET includes representations for planning problem context, goal specification, plan, plan task, and plan task description. The PLANET ontology can be used to retrieve data related to planning tasks and the plans generated (2/10 competency questions supported; `C4`, `C6` from Sect. 4.1). However, PLANET does not include representations for some entities commonly associated with planning domains, such as resources and time. Our planning ontology draws inspiration from PLANET and appends more metadata for planner improvement.

In [12], a domain-independent approach is presented that advances the state of the art by augmenting the knowledge of a planning task with pertinent goal opportunities. The authors demonstrate that incorporating knowledge obtained from an ontology can aid in producing better-valued plans, highlighting the potential for planner enhancement using more tuning parameters, which are captured in our planning ontology. The CARESSES ontology [13] is another significant development in planning-oriented ontologies, focusing on cultural competence in socially assistive robots for elderly care. Our work incorporates aspects from this ontology, specifically the concepts of `Action` and `Parameter`. The CARESSES ontology can be used to retrieve information about the actions in a domain (2/10 competency questions supported; `C3`, `C7` from Sect. 4.1).

The PROV-O ontology [14] provides a framework for representing provenance information, detailing the origins and transformations of data. In [15], P-Plan is introduced as an extension of the PROV-O ontology for modeling scientific processes. P-Plan effectively represents the steps, sequences, and dependencies in experimental workflows. However, its design primarily targets scientific investigations, limiting its direct applicability to automated planning (3/10 competency questions supported; `C4`, `C6`, `C7` from

Sect. 4.1). We have adapted the `Plan` concept from P-Plan to better suit the iterative and conditional nature of planning activities. We also reused the `State` concept from lifecycle ontology [16].

In our current work, we extended our preliminary work presented at a non-archival workshop [8]. Specifically, we have enhanced the ontology by introducing more detailed relationships and classifications within the domain, planner, and problem categories. The new ontology now includes refined subclasses for state, planning problems, and parameter types, along with more explicit connections between actions, preconditions, and effects. We have also incorporated concepts from existing ontologies within the Plan category (more details are provided in Sect. 4). Furthermore, we have added data properties of plan cost and plan explanation to support plan explanation generation. Additionally, we have standardized the terminology for the concepts and the properties used in our ontology. These additions aim to improve the overall clarity and functionality of the ontology, facilitating a better understanding and analysis of the planning processes. Furthermore, we include additional use cases of our ontology and provide experimental evaluations to support our findings.

4 Planning ontology

This section covers the construction of planning ontology to capture the essential details of automated planning. We will discuss the considerations, challenges, benefits, and limitations of using ontologies for automated planning, to provide a better understanding of how they can improve the efficiency and effectiveness of automated planning systems.

4.1 Competency questions

Competency questions for an ontology are focused on the needs of the users who will be querying the ontology. These questions are designed to help users explore and understand the concepts and relationships within the ontology, and to find the information they need within the associated knowledge base. By answering these questions, the ontology can be better scoped and tailored to meet the needs of its users.

We, in consultation with the domain experts, designed the following competency questions to model an ontology to represent the general aspects of classical Automated Planning. SPARQL queries for each of these questions can be found at our GitHub Repository [19].

- C1: What are the different types of planners used in automated planning?
- C2: What is the relevance of planners in a given problem domain?
- C3: What are the available actions for a given domain?
- C4: What problems in a domain satisfy a given condition?
- C5: What are all the requirements a given domain has?
- C6: What is the cost associated with generating a plan for a given problem?
- C7: How many parameters does a specific action have?
- C8: What planning type does a specific planner belong to?
- C9: What requirements does a given planner support?
- C10: What are the different parameter |types| present in a domain?

4.2 Design

We followed an iterative approach to develop the Planning Ontology. The process began with discussions with domain experts to gather insights. From there, we identified relevant classes, properties, and potential axioms, which were then incorporated into the ontology. Afterward, we used a reasoner to check for consistency. The ontology was evaluated against the competency questions and two specific use cases. If any competency questions or use case requirements were not met, the process was repeated, refining the ontology in each iteration until all criteria were satisfied. A preliminary version of this ontology, demonstrating its early development stages, was presented in a non-archival workshop [8].

Figure 2 shows an ontology that aims to encompass the various concepts of automated planning separated into categories of |Domain|, |Problem|, |Plan|, and |Planner|. The ontology for automated planning is composed of 19 distinct classes and 25 object properties. These classes and properties are designed to represent the various elements of the automated planning domain and its associated problems. In the design of our ontology, all axioms are formulated using Description Logic [20], providing a formal and expressive framework for representing and reasoning about the concepts and relationships within our domain.

4.2.1 Domain

The Domain category in our ontology comprises the characteristics of the AI planning domain through several classes. These include `PlanningDomain - DomainRequirement`, detailing domain modeling; `ParameterType`, defining parameter varieties in a typed domain; `DomainPredicate`, encompassing applicable predicates; `DomainConstant`, representing invariant constants; and `Action`, for domain operations. `Action` class is further linked with `ActionPrecondition`, `ActionEffect`, and `Parameter`. This structured approach aids applications like algorithm design, planner optimization, and macro learning in domain-specific contexts.

The `PlanningDomain` conceptualization is articulated through axioms to represent fundamental elements of planning scenarios. Axiom 1 signifies that every planning

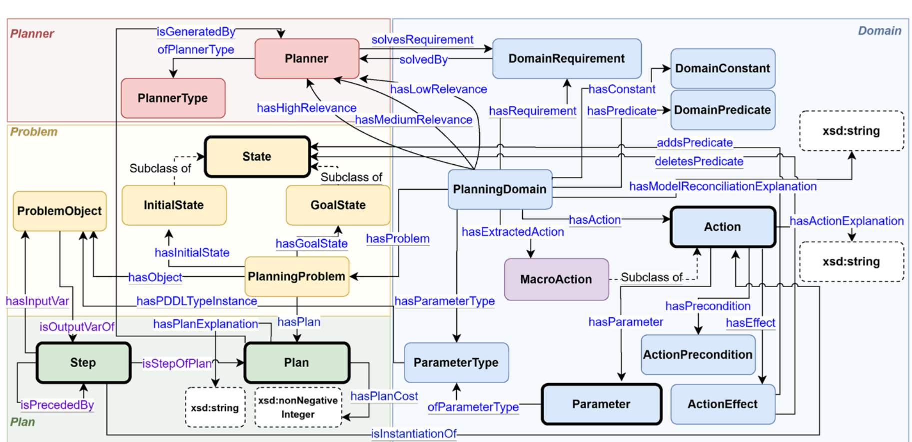

Fig. 2 An illustrative overview of the planning ontology, segmented into categories that encapsulate the core concepts of automated planning: domain, problem, plan, and planner performance. Each category is distinctly represented by colored rectangles. Classes with thick outlines denote concepts that have been adapted or re-used from existing ontologies. The data properties `hasPlanExplanation`, `hasActionExplanation`, and `hasModelReconciliationExplanation` [17, 18] help in providing explanations for user queries

domain entails certain actions. Actions are fundamental to planning as they represent the steps or decisions that can be taken to transform a state within the domain. Predicates are essential for defining the states within a planning domain. Axiom 2 ensures that each domain includes predicates to represent these states, facilitating the definition of preconditions and effects of actions. Axiom 3 states that every planning domain possesses certain defined requirements. Requirements in AI Planning are necessary to define various types of domain modeling, such as conditional effects and numeric fluents. Such specifications are not only essential for characterizing the domain but also serve as a criterion to assess whether a planner is compatible with and can support these specific domain modeling features.

$$0026; \mathrm{PlanningDomain} \sqsubseteq \exists \mathrm{hasAction.Action} \tag{1}$$

$$0026; \mathrm{PlanningDomain} \sqsubseteq \exists \mathrm{hasPredicate.DomainPredicate} \tag{2}$$

$$0026; \mathrm{PlanningDomain} \sqsubseteq \exists \mathrm{hasRequirement.DomainRequirement} \tag{3}$$

The `Action` class is characterized by its effects, a fundamental aspect of planning. Axiom 4 addresses the transformative nature of actions in a planning domain. Understanding the effects of actions is essential for planning algorithms to predict and evaluate the outcomes of different action sequences.

$$\mathrm{Action} \sqsubseteq \exists \mathrm{hasEffect.ActionEffect} \tag{4}$$

Axioms 5 and 6 capture the dynamics of how actions can add or delete predicates in a state, emphasizing the mutable nature of states within the planning domain. This depiction is essential for accurately modeling the consequences and feasibility of actions in AI Planning.

$$0026; \mathrm{ActionEffect} \sqsubseteq \exists \mathrm{addsPredicate.State} \tag{5}$$

$$0026; \mathrm{ActionEffect} \sqsubseteq \exists \mathrm{deletesPredicate.State} \tag{6}$$

4.2.2 Problem

The Problem category of the ontology includes classes that represent specific problems within a given domain. These classes are designed to capture the details of a particular problem, such as the |Objects| defined in the problem, which is an instance of different *types* defined in the planning domain, the |Initial State| of the problem, and the |Goal State| which are a subclass of the parent class |State| which is a state description of the given domain.

The axioms defined for `PlanningProblem` conceptualized the key aspects of a planning problem. Axiom 7 indicates that each planning problem is defined with a specific `GoalState`, which is the desired outcome or objective of the problem. Axiom 8 asserts that each planning problem also has a defined `InitialState`, which provides the starting conditions and context for the planning process. Lastly, Axiom 9 identifies the `Objects` present within a planning problem, denoting the various entities that are subject to manipulation or consideration during the course of planning. Finally, the axiom 10 underscores that every planning problem includes a potential plan or series of actions that lead to the goal state.

$$0026; \text{PlanningProblem} \sqsubseteq = 1\text{hasGoalState.GoalState} \tag{7}$$

$$0026; \text{PlanningProblem} \sqsubseteq = 1\text{hasInitialState.InitialState} \tag{8}$$

$$0026; \text{PlanningProblem} \sqsubseteq \exists\text{hasObject.ProblemObject} \tag{9}$$

$$0026; \text{PlanningProblem} \sqsubseteq \exists\text{hasPlan.Plan} \tag{10}$$

4.2.3 Plan

The Plan category of the ontology includes classes that represent the sequence of actions that must be taken to solve a given problem. The concepts in this category are adapted from the P-Plan ontology [15]. The |Plan| class captures the knowledge about the plans that planners generate for specific problems. The plan cost for each plan is a data property (non-negative integer) of the |Plan| class. This enables planners to be compared based on the quality of the plans they generate and the cost of those plans. The |Step| class from [15] stores each step of the plan.

The axioms defined for the `Plan` category outline the essential features of plans in the AI planning process. Axiom 11 mandates that each plan must have an associated plan cost, precisely quantified as a non-negative integer. This is crucial for evaluating and comparing the efficiency of different plans. Axiom 12 establishes that every plan is generated by some planner, connecting each plan to its generator and allowing for an understanding of the planning process and the assessment of various planners. Axiom 13 asserts that every `Step` is part of some `Plan`. This establishes a clear hierarchical relationship between steps and plans, ensuring that each individual step can be traced back to the larger plan it contributes to. This is important for understanding the structure and sequence of actions within a plan. Axiom 14 states that each `Step` must have an associated input variable that is a `ProblemObject`. This connects each step to the specific elements it operates on, providing a detailed representation of how the steps interact with the problem's components.

$$0026; \text{Plan} \sqsubseteq = 1\text{hasPlanCost.xsd:nonNegativeInteger} \tag{11}$$

$$0026; \text{Plan} \sqsubseteq \exists\text{isGeneratedBy.Planner} \tag{12}$$

$$0026; \text{Step} \sqsubseteq \exists\text{isStepOf.Plan} \tag{13}$$

$$0026; \text{Step} \sqsubseteq \exists\text{hasInputVar.ProblemObject} \tag{14}$$

4.2.4 Planner

The Planner category of the ontology includes classes that capture the details of the planner, planner type, and the planner performance from previous IPCs. Specifically, |Planning Domain| relevance to a |Planner| is classified based on the percentage of problems they have successfully solved, which is then categorized into three levels of relevance to the planner: *low*, *medium*, and *high*. By incorporating this information into the ontology, planners can be evaluated based on their performance in different planning domains, and more informed decisions can be made. In addition, this information can be used to guide the development of new planners and to evaluate their performance against established benchmarks.

The axioms defined for the `Planner` category provide a foundation for understanding and assessing the capabilities of planners in the AI planning domain. Axiom 15 classifies planners into different types based on their characteristics or strategies, enabling a nuanced understanding of various planning approaches. Axiom 16 links planners with the specific domain requirements they can solve, highlighting their applicability in different planning scenarios.

$$0026; \mathrm{Planner} \sqsubseteq \exists \mathrm{ofPlannerType}.\mathrm{PlannerType} \tag{15}$$

$$0026; \mathrm{Planner} \sqsubseteq \exists \mathrm{solvesRequirement}.\mathrm{DomainRequirement} \tag{16}$$

4.3 Accessing planning ontology

We have taken various measures to ensure that our planning ontology follows the FAIR principles [21] of being Findable, Accessible, Interoperable, and Reusable. To assist users in exploring and utilizing our ontology, we have made it accessible through a persistent URL [22] and our GitHub repository [19]. Our repository contains ontology model files, mapping scripts, and utility scripts that extract information from PDDL domains and problems into an intermediary JSON format and add the extracted data as triples using our model ontology, creating a knowledge graph. We provide sample SPARQL queries that address the ontology's competency questions mentioned earlier. Moreover, our ontology documentation, which is accessible through the GitHub repository, provides a comprehensive overview of the ontology's structure, concepts, and relations, including ontology visualization. This documentation serves as a detailed guide for users to comprehend the ontology's applications in the automated planning domain. We also provide the scripts and results from the ontology evaluation, which are presented as use cases of our ontology in later sections, in our repository, along with accompanying documentation.

FAIRness details Beyond ensuring basic findability, our ontology aligns with several specific FAIR principles: (i) Findable (F1–F3): Our ontology is assigned a persistent identifier (PURL) and described with rich, searchable metadata using standard vocabularies such as DCAT and `dcterms`; (ii) Accessible (A1): all ontology files, documentation, and example queries are openly available through the GitHub repository under an open license, with metadata preserved for long-term access; (iii) Interoperable (I1–I3): the ontology reuses existing vocabularies such as PROV-O and P-Plan, and adopts W3C-recommended RDF/OWL formats to facilitate machine interoperability; (iv) Reusable (R1–R1.1): a clear license (Creative Commons Attribution 4.0) and detailed provenance are provided to support community reuse and extension. Certain principles, such as F4 (registration in community metadata repositories), are not yet fully satisfied but are part of our planned work to integrate the ontology with public registries such as FAIRsharing.

Furthermore, our commitment includes a proactive approach to constantly updating and refining the ontology. This involves periodic updates and community-driven modifications, ensuring its continuous alignment with evolving standards and practices in the field of automated planning .

5 Usecase 1: identifying most promising planner

One of the major challenges in the field of artificial intelligence (AI) is the automated selection of the most promising planner for a given planning domain. This challenge arises due to the vast number of available planners and the diversity of planning domains. The traditional way to select a planner is to experiment with various search algorithms and heuristics and settle on an appropriate combination as seen in IPC competitions. To address this challenge, we now present a new approach by using our planning ontology, presented in Figure 3, to represent the features of the planning domain and the capabilities of planners.

Our Planning Ontology captures the relationship between the Planning Domain and the Planner by indicating the relevance of a planner to a specific domain. We made use of data acquired from International Planning Competitions (IPCs) to furnish specific details regarding the relevance of planners. The IPC results provide us with relevant details on the planners that took part in the competition and the domains that were evaluated during that particular year. This information includes specifics on how each planner performed against all the domains that participated.

To show the usage of extracting the most promising planners for a given domain, we have used IPC data (optimal track) [23]. A relevance relation of either *low*, *medium*, or *high* was assigned to each planner based on the percentage of problems, *low*-[below 35%], *medium*-[35% to 70%], *high*-[70% and above], they solved in a given domain. In this experiment, we consider that the experimental environment has four planners: Fast Downward [Stone Soup 1, A* + LM-Cut, A* + Merge and Shrink], and BJOLP [24]. We evaluate 3 problem instances of each domain (mentioned in Section 2.2). We have two policies—Random Policy and Ontology Policy—for assigning planners to solve these problems. (1) Random Policy: To solve each problem instance, this policy selects a random planner from the available planners. (2) Ontology Policy: To solve each problem instance, this policy extracts the information on the best planner, having `hasHighRelevance` object property, for the problem domain from the ontology populated with IPC-2011 data.

This experiment is designed to demonstrate how the structured knowledge captured in the Planning Ontology can be effectively used to reason about planner-domain relationships and identify the most suitable planner for a given domain. Rather than aiming for state-of-the-art performance, the focus is on showing how ontology-driven reasoning can automate the planner selection process by leveraging semantic relationships (`hasHighRelevance`). Additionally, the Appendix (Table A1) highlights extended results showing our framework evaluated on two randomly selected domains from each IPC edition, further illustrating its general applicability across multiple planning benchmarks.

```
Q: "Which is the best planner for blocksworld domain?"
SELECT ?planner
WHERE {
    po:blocksworld po:hasHighRelevance ?planner.
}
```

Table 1 presents the results of our evaluation, indicating the average number of nodes expanded, during the search, to find a solution and plan cost for each policy in a given domain. Additional results for other domains can be found in Appendix A.1 Table A1. The table provides a comprehensive summary of the performance of different planners

Table 1 Demonstrating the effectiveness of two different policies for problem-solving

Domain	Ontology policy		Random policy	
	Avg. Exp. ± Std.	Avg. Cost ± Std.	Avg. Exp. ± Std.	Avg. Cost ± Std.
Scanalyzer	**8.6k ± 0**	20 ± 0	8.7k ± 0	20 ± 0
Elevators	**1.5k ± 0**	52 ± 0	65k ± 0	52 ± 0
Transport	170k ± 0	491 ± 0	**130k ± 0**	491 ± 0
Parking*	**370k ± 0**	18 ± 0	490k ± 0	17 ± 0
Woodworking	**2.0k ± 0**	211 ± 0	20k ± 0	211 ± 0
Floortile**	280k ± 0	54 ± 0	**2.1k ± 0**	49 ± 0
Barman	**1.3M ± 0**	90 ± 0	5.8M ± 0	90 ± 0
Openstacks	**130k ± 0**	4 ± 0	140k ± 0	4 ± 0
Nomystery	1.7k ± 0	13 ± 0	1.7k ± 0	13 ± 0
Pegsol	**89k ± 0**	6 ± 0	100k ± 0	6 ± 0
Visitall	5 ± 0	4 ± 0	5 ± 0	4 ± 0
Tidybot**	**1.2k ± 0**	17 ± 0	3.4k ± 0	33 ± 0
Parcprinter	540 ± 0	441k ± 0	**420 ± 0**	441k ± 0
Sokoban	**9.7k ± 0**	25 ± 0	160k ± 0	25 ± 0

This table compares the average nodes expanded (Avg. Exp.) during search and the resulting plan cost (Avg. Cost) for the Ontology Policy and Random Policy

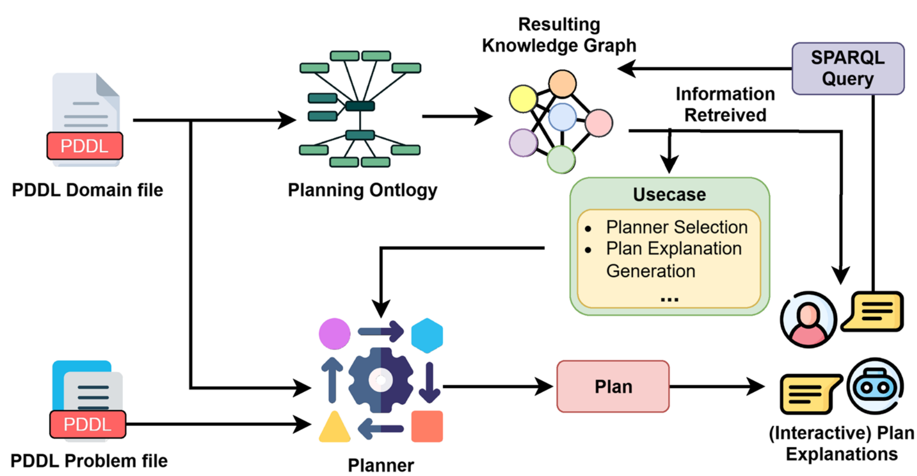

Fig. 3 Workflow diagram illustrating the integration of the Planning Ontology with AI Planning system, which supports use cases of Most Promising Planner Selection and generating Plan Explanations with the help of generated Knowledge graph

in terms of their efficiency and effectiveness. An ideal planner is expected to generate a solution with low values for both of these metrics. The *Ontology Policy*, designed to select the most promising planner for a given domain outperformed the *Random Policy* in terms of the average number of nodes expanded to find a solution. Moreover, the *Random Policy* failed to solve problems in the parking (1 out of 3), floortile (2 out of 3), and tidybot (2 out of 3) domains, which highlights the limitations of choosing a planner randomly. But if a domain is easily solvable by relevant planners that can tackle them, *Random Policy* may still do well.

The query below identifies the best planner for the blocksworld domain by prioritizing planners with high relevance. If none exist, it selects medium-relevance planners, and as a last resort, low-relevance planners. The `COALESCE` function ensures a single, most suitable planner is chosen, enabling efficient and transparent planner selection.

```
Q: "Which is the best planner for blocksworld domain?"
SELECT ?planner
WHERE {
  {
    po:blocksworld po:hasHighRelevance ?highRelevancePlanner
  }
  UNION
  {
    FILTER NOT EXISTS { po:blocksworld po:hasHighRelevance ?
        highRelevancePlanner }
    po:blocksworld po:hasMediumRelevance ?mediumRelevancePlanner
  }
  UNION
  {
    FILTER NOT EXISTS { po:blocksworld po:hasHighRelevance ?
        highRelevancePlanner }
    FILTER NOT EXISTS { po:blocksworld po:hasMediumRelevance ?
        mediumRelevancePlanner }
    po:blocksworld po:hasLowRelevance ?lowRelevancePlanner
  }
}
BIND(COALESCE(?highRelevancePlanner, ?mediumRelevancePlanner, ?
    lowRelevancePlanner) AS ?planner)
LIMIT 1
```

6 Usecase 2: explanation generation

In the field of automated planning, generating clear and comprehensible explanations continues to be a significant challenge. While contemporary techniques excel at plan production, they often fall short in offering human-understandable explanations and justifications for these plans. This deficit can hinder trust and collaboration, especially in contexts demanding fluid human-AI interactions. The inherent complexities of planning problems underscore the imperative for explainable planning. The prevailing literature delineates five primary explanation categories pertinent to automated planning [25]—Plan Explanation [26], Verbalization [27], Model Reconciliation [17, 18], Explaining Specific Actions and Contrastive Explanations [28]. We can support these by using the causal relationships represented in the ontology, analysis of the plan from a plan validator such as VAL [29], and a template-based text generator. In the future, we plan to augment explanations with automatically generated metadata about plans, e.g., plan structure [30] and other avenues of using semantics identified by [31] to provide context for richer explanations.

Our ontology-driven approach, presented in Figure 3, uses semantic web technologies to generate diverse explanation types by encoding planning domain knowledge, action semantics, and plan structures within the ontology. This enables the extraction of contextually rich explanations through SPARQL queries. We support three fundamental categories of planning explanations, as outlined in Table 2, ranging from high-level plan summaries to detailed justifications of individual actions. The following section expands on each category including a user question and the corresponding SPARQL query for our Planning Ontology. Note that the SPARQL queries provided below omit prefix declarations. In practice, appropriate prefixes (e.g., `po:`, `rdf:`) should be included at the beginning of each query.

Plan Explanation is a crucial component in making automated planning systems more transparent and accessible. This category encompasses various approaches to translate

Table 2 The following table shows examples of answers generated from the information retrieved using the Planning Ontology for different explanation types, corresponding to the planning problem depicted in Fig. 1

Type of explanation	Description	Example question	Example response	User survey result (out of 5)
Plan explanation	Involves translating planner outputs (e.g., PDDL plans) into forms that humans can easily understand	*Can you explain the plan to achieve the goal configuration in simple terms?*	Removed block 2 from block 1, placed block 2 on the table, picked up block 3 to stack on block 1, then stacked block 2 on block 3 to achieve the goal configuration.	4.20
Explaining specific actions	Explains why a specific action is taken in a plan	*Why did we unstack block 2 from block 1 as the first step?*	Unstacked block 2 from block 1 to free it; placed block 2 on the table for clear rearrangement; picked up block 3 to position above block 1; stacked block 2 on block 3 to finalize the desired configuration.	3.93
Explaining non-selection of specific actions	When a planner's decision is contrasted with an alternative suggested by a human, an explanation should demonstrate why the alternative action was not chosen.	*Why didn't the planner stack block 3 on block 1 before moving block 2?*	The action "stack block 3 on block 1" was not selected because precondition "clear block 1" was not satisfied; action "unstack block 2 from block 1" was necessary first to satisfy the precondition; directly stacking block 3 on block 1 would violate the constraint "only one block can be on another block at a time".	3.73

complex PDDL plans into human-comprehensible formats, and bridges the gap between machine efficiency and human understanding. This approach facilitates better human-AI collaboration, as it allows non-expert users to quickly grasp the essence of a computed plan without the need for technical details.

High-Level Plan Summary provides an overview of the entire plan, explaining its validity and how it achieves the goal. It offers a broad perspective on the plan's structure and purpose.

```
Q: "Why is this plan valid for achieving the goal?"
SELECT ?explanation
WHERE {
    ?plan a po:plan.
    ?plan po:hasPlanExplanation ?explanation .
}
```

The query above retrieves the plan explanation associated with a specific plan using the `hasPlanExplanation` data property (ref. Fig. 2). The retrieved explanation provides a high-level summary of why the plan is valid and how it achieves the goal, offering users a quick understanding of the plan's overall strategy.

Natural Language Generation (NLG) for explaining plan steps involves translating the formal representation of actions, preconditions, and effects into natural language descriptions. Similar to the work in [26], where the authors focused on verbalizing task plans through semantic tagging of actions and predicates, our approach aims to enhance the understandability of planning systems. While the ontology itself remains independent of the labels, incorporating meta-data within the labels allows for the generation of natural language explanations. This integration supports the creation of user-friendly, contextually rich descriptions of planning processes. The query below extracts the parameters, precondition labels, and effect labels of a specific action (in this case, `'pick-up')`. By mapping these technical details to natural language templates, we can generate human-readable explanations.

```
Q: "Can you describe the 'pick-up' action in simple terms?"
SELECT ?param ?preconditionLabel ?effectLabel
WHERE {
    po:pick-up po:hasParameter ?param ;
                po:hasPrecondition ?preCondition ;
                po:hasEffect ?effect .
    ?preCondition rdf:label ?preconditionLabel .
    ?effect rdf:label ?effectLabel .
}
```

Explaining Specific Actions focuses on justifying why a particular action was chosen at a specific point in the plan, often related to the overall goal or the current state of the world. In complex planning scenarios, understanding why a particular action was chosen can be crucial for trust and system optimization.

```
Q: "Why was the 'pick-up' action chosen at step 3 of the plan?"
SELECT ?actionExplanations
WHERE {
    po:pick-up po:hasActionExplanation ?actionExplanations.
}
```

This query retrieves the action explanation for an action using `hasActionExplanation` data property (ref. Fig. 2). This approach allows users to understand not just what the plan does, but why each step is necessary.

Explaining Non-Selection of Specific Actions addresses why certain actions were not chosen, which can be crucial for understanding the planner's decision-making process and validating the optimality of the plan. Furthermore, providing contrastive explanations with the chosen action enhances the system's accountability and helps users understand the trade-offs considered during the planning process. The query below allows us to extract and compare the preconditions and effects of two different actions. By analyzing this information, we can generate explanations that highlight why one action was preferred over another. This might involve identifying unsatisfied preconditions in the resulting state, comparing the effects in relation to the goal state, or explaining how the chosen action better optimizes certain metrics (e.g., plan length, and resource usage).

```
Q: "Why did you perform (pick-up b3) instead of unstack in step
    3?"
SELECT ?action ?preconditionLabel ?effectLabel
WHERE {
  ?action po:hasParameter ?param .
  FILTER (?action IN (po:pick-up, po:unstack))

  ?action po:hasPrecondition ?preCondition .
  ?preCondition rdf:label ?preconditionLabel .

  ?action po:hasEffect ?effect .
  ?effect rdf:label ?effectLabel .
}
ORDER BY ?action
```

In practice, basic queries can be combined with more complex logic to retrieve appropriate ontology information. This retrieved information is then processed to generate natural language responses, ensuring that the data is presented in a meaningful and coherent format for the user. (as shown in Table 2).

User Survey Design: To gain insights into user preferences and the perceived effectiveness of different explanation types, we conducted a preliminary survey using two IPC domains: Blocksworld and Sokoban. The participants were graduate students and researchers in computer science, familiar with the fundamentals of automated planning and reasoning.

Participants were presented with one problem instance from each domain. For each instance, they were given a short description, an image depicting the initial and goal configurations, and ontology-generated explanations corresponding to the three explanation categories described in Table 2: *action-level*, *predicate-level*, and *plan-level* explanations. These explanations were automatically generated using templates and knowledge extracted from the Planning Ontology, ensuring consistent and semantically grounded responses across both domains.

To assess the explanations, participants were asked to respond to the two questions for each explanation category: *To what extent did the explanation help you understand the rationale behind the plan?* and *How would you rate the clarity of the explanation provided for the plan?* These questions were conducted on a 1–5 Likert scale [32] and aimed to evaluate both the utility and clarity of the ontology-generated explanations. The collected responses helped evaluate how effectively the ontology supports the generation of human-understandable explanations in planning contexts.

User Survey Results: This is a preliminary study, and we collected 10 responses. We plan to conduct a more extensive survey in future to further validate our findings. To quantify user preferences and the perceived effectiveness of each explanation type, we calculate the average scores obtained from the survey (presented in Table 2). These scores were calculated based on responses to two types of questions for each explanation category across both domains, and they represent the participants' evaluations of how helpful and clear each explanation type was.

The “Plan Explanation" category received the highest average score of 4.20, indicating that users found this form of explanation to be the most clear and helpful overall. The “Explaining Specific Actions" category followed closely with an average score of 3.93, suggesting that users also appreciated explanations that clarified why certain actions were selected within the plan. In contrast, the “Explaining Non-Selection of Specific Actions" category received a slightly lower average score of 3.73. This indicates that while participants still found this type of explanation useful, it was perceived as less effective or comprehensible compared to the other two categories.

7 Conclusion

In this work, we build and share a planning ontology that provides a structured representation of concepts and relations for planning, allowing for efficient extraction of domain, problem, and planner properties. The ontology's practical utility is demonstrated in identifying the best-performing planner for a given domain and showcasing the generation of comprehensive plan explanations. Additionally, the ontology enables further applications, such as macro-action generation [8]. In the future, we aim to conduct a comprehensive user study to evaluate the usefulness of the generated explanations for user satisfaction. Standardized benchmarks from IPC domains and planners offer an objective and consistent approach to evaluating planner performance, enabling rigorous comparisons in different domains to identify the most suitable planner. The planning ontology can aid researchers and practitioners in automated planning, and its use can simplify planning tasks and boost efficiency. As the field of AI planning continues to evolve, planning ontology can play a crucial role in advancing the state-of-the-art while leveraging the past.

Future work could explore the use of a mixed reasoning strategy that combines the structured, top-down approach of ontologies with the dynamic, bottom-up capabilities of Large Language Models (LLMs) [33]. This approach can be particularly effective in contexts like LLMs, which have shown promise for automated planning [34]. Furthermore, our ontology, with its specific data properties for storing action explanations, can be leveraged to enhance this hybrid model. Similar to the work in [35], where iterative prompting strategies are employed, providing feedback of observations from a Plan Validator [29], to help LLMs reason better, the information retrieved from the ontology can be used to enhance prompts with appropriate domain information and relevant context, improving their ability to generate accurate and coherent explanations. This blend of ontology-based clarity and LLM-driven adaptability can help coordinate actions and explain them in a transparent and informative way.

Limitations: The effectiveness of the different use cases presented using our Planning Ontology depends on the data available within the ontology, and its performance may be limited if the data is incomplete or disproportionately represents certain domains or planners. Additionally, the template-based explanations, while useful, may not meet all users' needs for personalized and context-sensitive responses.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1007/s44248-025-00093-9.

Supplementary file 1.

Acknowledgements

This work was partially inspired by support from the Applied Research Laboratory for Intelligence and Security (ARLIS) at the University of Maryland. We also acknowledge partial funding from the University of South Carolina and the JP Morgan Research Faculty Award to Biplav Srivastava.

Author contributions

Bharath Muppasani, Nitin Gupta, Vishal Pallagani wrote the main manuscript text. Bharath Muppasani worked on Section 1, Section 4, Section 5, and the Appendix. Nitin Gupta worked on Section 6 - Usecase 2: Explanation Generation and running experiments. Vishal Pallagani worked on Section 2 & Section 3 of the manuscript. Biplav Srivastava, Raghava Mutharaju, Michael Huhns, and Vignesh Narayanan worked on Section 1, Section 4, and Section 7. All authors reviewed the manuscript.

Data availability

The ontology developed in the research work is published as a persistent URL - https://purl.org/ai4s/ontology/planning Additional data and code files supporting the research work have been published in a public GitHub repository - https://github.com/ai4society/planning-ontology

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 26 June 2025 / Accepted: 14 November 2025

Published online: 26 November 2025

References

1. Ghallab M, Nau D, Traverso P. Automated planning: theory and practice. San Francisco: Elsevier; 2004.
2. SPARK: Scheduling and planning applications workshop. 2024. https://icaps23.icaps-conference.org/program/workshops/spark/. https://ccia.ugr.es/~lcv/SPARK/.
3. Karpas E, Magazzeni D. Automated planning for robotics. Ann Rev Contr Robot Auton Syst. 2020;3(1):417–39.
4. Behandish M, Nelaturi S, Kleer J. Automated process planning for hybrid manufacturing. Comput Aided Des. 2018;102:115–27.
5. Guarino N, Oberle D, Staab, S. What is an ontology? Handbook on ontologies. 2009;1–17.
6. Bihl TJ, Cox C, Machin T. Towards a taxonomy of planning for autonomous systems. In: 2019 IEEE National Aerospace and Electronics Conference (NAECON), 2019;74–79. https://doi.org/10.1109/NAECON46414.2019.9058324.
7. Gil Y, Blythe J. Planet: A shareable and reusable ontology for representing plans. In: AAAI Workshop on Representational Issues for Real-world Planning Systems. Website: Https://www.isi.edu/∼blythe/planet/. 2000;114.
8. Muppasani B, Pallagani V, Srivastava B, Mutharaju R. Building and using a planning ontology from past data for performance efficiency. In: Proceedings of the planning and ontology workshop (PLATO), co-located with the 33rd international conference on automated planning and scheduling (ICAPS'23). 2023.
9. Valente A, Russ T, MacGregor R, Swartout W. Building and (re) using an ontology of air campaign planning. IEEE Intell Syst Appl. 1999;14(1):27–36.
10. Žáková M, Křemen P, Železnỳ F, Lavrač N. Automating knowledge discovery workflow composition through ontology-based planning. IEEE Trans Autom Sci Eng. 2010;8(2):253–64.
11. Gayathri R, Uma V. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. ICT Express. 2018;4(2):69–74.
12. Babli M, Onaindia E, Marzal E. Extending planning knowledge using ontologies for goal opportunities. 2019. arXiv preprint arXiv:1904.03606.
13. Khaliq AA, Köckemann U, Pecora F, Saffiotti A, Bruno B, Recchiuto CT, Sgorbissa A, Bui H-D, Chong NY. Culturally aware planning and execution of robot actions. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2018;326–332. IEEE. http://caressesrobot.org/ontology.
14. Lebo T, Sahoo S, McGuinness D, Belhajjame K, Cheney J, Corsar D, Garijo D, Soiland-Reyes S, Zednik S, Zhao J. Prov-o: The prov ontology. W3C recommendation. 2013;30.
15. Garijo D, Gil Y. Augmenting prov with plans in p-plan: scientific processes as linked data. 2012. CEUR Workshop Proceedings. https://www.opmw.org/p-plan.owl.
16. Lifecycle Schema Vocabulary. http://purl.org/vocab/lifecycle/schema. Accessed 26 Sep 2024.
17. Chakraborti T, Sreedharan S, Zhang Y, Kambhampati S. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. 2017. arXiv preprint arXiv:1701.08317.
18. Krarup B, Krivic S, Magazzeni D, Long D, Cashmore M, Smith DE. Contrastive explanations of plans through model restrictions. J Artif Intell Res. 2021;72:533–612.
19. Ai4s: Planning ontology. https://github.com/ai4society/planning-ontology/. Accessed 26 Sep 2024.
20. Krötzsch M, Simančík F, Horrocks I. Description logics. IEEE Intell Syst. 2014;29:12–9.
21. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The fair guiding principles for scientific data management and stewardship. Sci Data. 2016;3:1–9.
22. AI4S planning ontology PURL. https://purl.org/ai4s/ontology/planning. Accessed 26 Sep 2024.
23. International Planning Competition 2011 - Deterministic Track. http://www.plg.inf.uc3m.es/ipc2011-deterministic/. Accessed 26 Sep 2024.
24. Fast Downward - IPC Planners. https://www.fast-downward.org/IpcPlanners. Accessed 26 Sep 2024.
25. Fox M, Long D, Magazzeni D. Explainable planning. 2017. arXiv preprint arXiv:1709.10256.
26. Canal G, Krivić S, Luff P, Coles A. Planverb: Domain-independent verbalization and summary of task plans. Proc AAAI Conf Artif Intell. 2022;36:9698–706.
27. Rosenthal S, Selvaraj SP, Veloso MM. Verbalization: Narration of autonomous robot experience. IJCAI. 2016;16:862–8.
28. Zehtabi P, Pozanco A, Bolch A, Borrajo D, Kraus S. Contrastive explanations of centralized multi-agent optimization solutions. Proc Int Conf Autom Plan Sched. 2024;34:671–9.
29. Howey R, Long D, Fox M. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In: 16th IEEE international conference on tools with artificial intelligence. 2004;294–301. IEEE.
30. Srivastava B, Vanhatalo J, Koehler J. Managing the life cycle of plans. In: AAAI. 2005;1569–1575.
31. Lécué F. On the role of knowledge graphs in explainable ai. Semantic Web. 2020;11(1):41–51.
32. Likert R. A technique for the measurement of attitudes. Archives of psychology. 1932.
33. Mittal S, Joshi A, Finin TW. Thinking, fast and slow: Combining vector spaces and knowledge graphs. 2017. ArXiv:1708.03310.
34. Pallagani V, Muppasani B, Murugesan K, Rossi F, Horesh L, Srivastava B, Fabiano F, Loreggia A. Plansformer: generating symbolic plans using transformers. 2022. In: On Arxiv At: arXiv:org/abs/2212.08681.
35. Stein K, Koller A. Autoplanbench:: Automatically generating benchmarks for llm planners from pddl. 2023. arXiv preprint arXiv:2311.09830.

Publisher's Note