Building a Plan Ontology to Represent and Exploit Planning Knowledge and Its Applications

Bharath Muppasani
bharath@email.sc.edu
University of South Carolina, USA

Nitin Gupta
niting@email.sc.edu
University of South Carolina, USA

Vishal Pallagani
vishalp@mailbox.sc.edu
University of South Carolina, USA

Biplav Srivastava
biplav.s@sc.edu
University of South Carolina, USA

Raghava Mutharaju
raghava.mutharaju@iiitd.ac.in
IIIT-Delhi, India

Michael N. Huhns
huhns@sc.edu
University of South Carolina, USA

Vignesh Narayanan
vignar@sc.edu
University of South Carolina, USA

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 consider the problem of automated planning, where the objective is to find a sequence of actions that will move an agent from an initial state of the world to a desired goal state. We hypothesize that given a large number of available planners and diverse planning domains, they carry essential information that can be leveraged to improve many ontology applications. We use open data on planning domains and planners to construct the most comprehensive planning ontology to date, based on supported competency questions, and 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.

Resource Type: Ontology, Knowledge Graph

PURL: https://purl.org/ai4s/ontology/planning
URL: https://github.com/BharathMuppasani/AI-Planning-Ontology

Keywords

Ontology, Automated Planning, Planner Selection, Explanation.

ACM Reference Format:
Bharath Muppasani, Nitin Gupta, Vishal Pallagani, Biplav Srivastava, Raghava Mutharaju, Michael N. Huhns, and Vignesh Narayanan. 2024. Building a Plan Ontology to Represent and Exploit Planning Knowledge and Its Applications. In *Proceedings of International Conference on Data Science and Management of Data (CODS COMAD '24).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

CODS COMAD '24, December 18–21, 2024, IIT Jodhpur, India

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Automated planning, where the objective is to find a sequence of actions that will transition an agent from the initial state of the world to a desired goal state, is an active sub-field of Artificial Intelligence (AI) [8]. 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 (see workshop series on applications called SPARK[21]). Currently, there are numerous planning domains, planners, search algorithms, and associated heuristics in the field of automated planning. Each planner, in conjunction with a search algorithm and heuristic, generates plans with varying degrees of quality, cost, and optimality. The empirical results available for various planning problems, ranked by planner performance and the heuristics used as available in the International Planning Competition (IPC), can provide valuable information to identify various tunable parameters to improve planner performance. Traditionally, improving planner performance involves manually curating potential combinations to identify the optimal planner configuration. However, there has been limited effort to model the available information in a structured knowledge representation, such as an ontology, to facilitate efficient reasoning and enhance planner performance.

To address the challenge of representing planning problems and associated information in a structured manner, we propose the most comprehensive ontology for AI planning - Planning Ontology. An ontology formally represents concepts and their relationships [10], which enables systematic analysis of planning domains and planners. The proposed ontology captures the features of a domain and the capabilities of planners, facilitating reasoning with existing planning problems, identifying similarities, and suggesting different planner configurations. 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 planning, several attempts have been made to create ontologies to enhance the understanding of planners' capabilities. For instance, Plan-Taxonomy [2] introduced a taxonomy that aimed to explain the functionality of planners. Additionally,

authors in [9] present a comprehensive ontology called PLANET, which represents plans in real-world domains and can be leveraged to construct new applications. Nonetheless, the reusability of PLANET is limited as it is not open-sourced and also supports a subset of capabilities of our proposed ontology (see capabilities discussion in Section 4.1). Consequently, researchers face difficulty in extending or replicating the ontology.

This paper outlines our methodology for constructing an ontology to represent *classical*[1] AI planning domains, leveraging information obtained from the IPC and supporting practical usability requirements like explainability. 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 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 [18]. Our contributions are at the intersection of ontologies and AI planning and can be summarized as follows.

- **Building Planning Ontology**: We develop the most comprehensive 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, also known as AI planning, is the process of finding a sequence of actions that will transform an initial state of the world into a desired goal state [8]. It involves constructing a plan or a sequence of actions that will achieve a specified objective while respecting any constraints or limitations that may be present. Formally, automated planning can be 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. A problem is 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 [8]. 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 an initial state can be transformed into a goal state by executing 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.
(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.

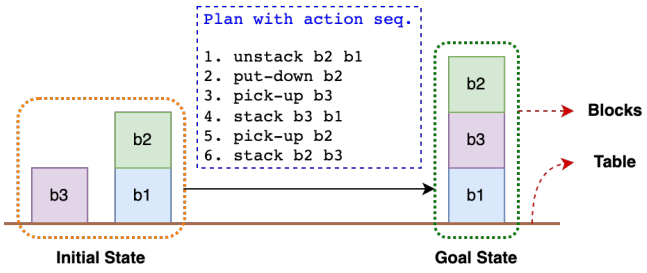

Figure 1: Demonstration of automated planning problem with blocksworld domain example

[1]Explained in preliminaries Section 2.1.

(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 (IPC)

The International Planning Competition (IPC) is essential for evaluating planning systems and promoting new methodologies. It includes multiple tracks, such as classical and probabilistic planning, with benchmarks assessing plan quality, length, and runtime. IPC results highlight the strengths and weaknesses of various systems. The benchmarks from IPC are ideal for crafting a planning-related ontology, encapsulating the domain's breadth and planners' challenges.

For our experiments, we used the 14 domains from IPC-2011, including scanalyzer, elevators, transport, parking, woodworking, floortile, barman, openstacks, nomystery, pegsol, visitall, tidybot, parcpinter, 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.

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 [24], 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 [26], 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, [7] suggests that knowledge reasoning approaches can draw new conclusions in non-deterministic contexts and assist with dynamic planning. In [9], 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 Section 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 [1], 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 [12] 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 Section 4.1) The PROV-O ontology [15] provides a framework for representing provenance information, detailing the origins and transformations of data. In [6], 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 Section 4.1). We have adapted the `Plan` concept from P-Plan to better suit the iterative and conditional nature of planning activities.

In our current work, we extended our preliminary work presented at a non-archival workshop [18]. 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 Section 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.

Table 1: Concepts reused from various ontologies

Concept	Ontology
Action	http://caressesrobot.org/ontology [12]
Parameter	http://caressesrobot.org/ontology [12]
Plan, Step	https://www.opmw.org/p-plan.owl [6]
State	http://purl.org/vocab/lifecycle/schema

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[2].

- 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

An ontology is a formal and explicit representation of concepts, entities, and their relationships in a particular domain. In this case, ontology is concerned with the domain of automated planning, which refers to the process of generating a sequence of actions to achieve a particular goal within a given set of constraints. The ontology aims to provide a structured framework for organizing and integrating knowledge about this domain, which can be useful in various applications, such as designing planning algorithms, extracting best-performing planners given a domain, or learning domain-specific macros.

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 [14], 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 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.

$$\texttt{PlanningDomain} \sqsubseteq \exists \texttt{hasAction.Action} \tag{1}$$

$$\texttt{PlanningDomain} \sqsubseteq \exists \texttt{hasPredicate.DomainPredicate} \tag{2}$$

$$\texttt{PlanningDomain} \sqsubseteq \exists \texttt{hasRequirement.DomainRequirement} \tag{3}$$

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.

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

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.

$$\texttt{ActionEffect} \sqsubseteq \exists \texttt{addsPredicate.State} \tag{5}$$

$$\texttt{ActionEffect} \sqsubseteq \exists \texttt{deletesPredicate.State} \tag{6}$$

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

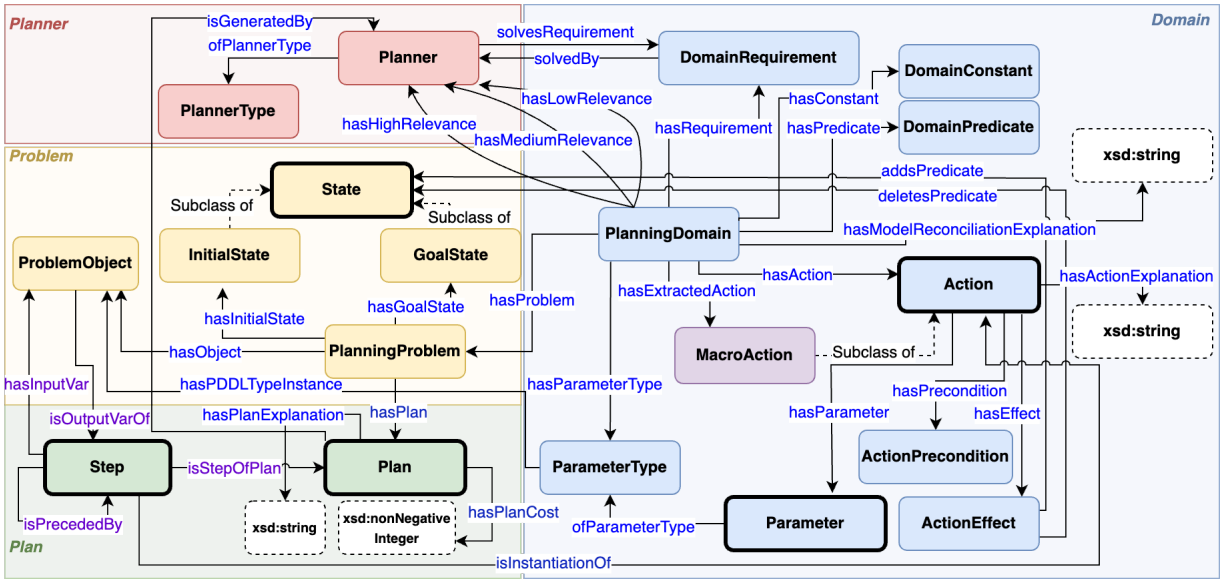

Figure 2: An illustrative overview of the planning ontology, segmented into categories that encapsulate the core concepts of automated planning: domain, problem, plan, and planner performance. Each category is distinctly represented by colored rectangles. Classes with thick outlines denote concepts that have been adapted or reused from existing ontologies. The data properties `hasPlanExplanation`, `hasActionExplanation`, and `hasModelReconciliationExplanation` [4, 13] help in providing explanations for user queries.

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.

$$\texttt{PlanningProblem} \sqsubseteq = 1\texttt{hasGoalState.GoalState} \quad (7)$$

$$\texttt{PlanningProblem} \sqsubseteq = 1\texttt{hasInitialState.InitialState} \quad (8)$$

$$\texttt{PlanningProblem} \sqsubseteq \exists\texttt{hasObject.ProblemObject} \quad (9)$$

$$\texttt{PlanningProblem} \sqsubseteq \exists\texttt{hasPlan.Plan} \quad (10)$$

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 [6]. 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 [6] 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.

$$\texttt{Plan} \sqsubseteq = 1\texttt{hasPlanCost.xsd:nonNegativeInteger} \quad (11)$$

$$\texttt{Plan} \sqsubseteq \exists\texttt{isGeneratedBy.Planner} \quad (12)$$

$$\texttt{Step} \sqsubseteq \exists\texttt{isStepOf.Plan} \quad (13)$$

$$\texttt{Step} \sqsubseteq \exists\texttt{hasInputVar.ProblemObject} \quad (14)$$

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

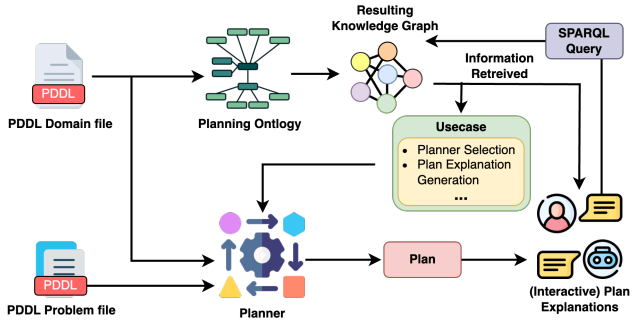

Figure 3: Workflow diagram illustrating the integration of the Planning Ontology with AI Planning system, which supports use cases of Most Promising Planner Selection and generating Plan Explanations with the help of generated Knowledge graph.

in 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.

$$\texttt{Planner} \sqsubseteq \exists \texttt{ofPlannerType.PlannerType} \tag{15}$$

$$\texttt{Planner} \sqsubseteq \exists \texttt{solvesRequirement.DomainRequirement} \tag{16}$$

4.3 Accessing Planning Ontology

We have taken various measures to ensure that our planning ontology follows the FAIR principles [25] 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[1] and our GitHub repository[2]. Our repository contains ontology model files, mapping scripts, and utility scripts that extract information from PDDL domains and problems into 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. 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.

[1]PURL - https://purl.org/ai4s/ontology/planning
[2]https://github.com/BharathMuppasani/AI-Planning-Ontology

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 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[3] (optimal track). A relevance relation of either *low*, *medium*, or *high* was assigned to each planner based on the percentage, *low*-below 35%, *medium*-35% to 70%, *high*-70% and above, of problems they solved in a given domain. In this experiment, we consider that the experimental environment has four planners available: Fast Downward Stone Soup 1[4], LM-Cut[4], Merge and Shrink[4], and BJOLP[4]. We evaluate 3 problem instances of each domain (mentioned in Section 2.2) with

[3]http://www.plg.inf.uc3m.es/ipc2011-deterministic/
[4]https://www.fast-downward.org/IpcPlanners

Table 2: Demonstrating the effectiveness of two different policies employed to choose a planner for problem-solving. Comparing the average nodes expanded during the search and the resulting plan cost for two policies.

Domain	Ontology Policy		Random Policy	
	Avg. Exp.	Avg. Cost	Avg. Exp	Avg. Cost
scanalyzer	**8,588**	20	8,706	20
elevators	**1,471**	52	64,541	52
transport	165,263	491	**132,367**	491
parking*	**367,910**	18	488,830	17
woodworking	**1,988**	211	19,844	211
floortile**	283,724	54	**2,101**	49
barman	**1,275,078**	90	5,816,476	90
openstacks	**132,956**	4	139,857	4
nomystery	1,690	13	1,690	13
pegsol	**89,246**	6	101,491	6
visitall	5	4	5	4
tidybot**	**1,173**	17	3,371	33
parcprinter	541	441,374	**417**	441,374
sokoban	**9,653**	25	156,600	25

2 policies for selecting planners to generate plans for each of these problem instances -

(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 for the problem domain from the ontology populated with IPC-2011 data.

Q: *"Which is the best planner for blocksworld domain?"*

```
SELECT ?planner
WHERE {
    po:blocksworld po:hasHighRelevance ?planner.
}
```

Table 2 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. The table provides a comprehensive summary of the performance of different planners 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 (SPARQL query shown above and in Appendix A.1), 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.

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 [5] - Plan Explanation [3], Verbalization [20], Model Reconciliation [4, 13], Explaining Specific Actions and Contrastive Explanations [27]. We can support these by using the causal relationships represented in the ontology, analysis of the plan from a plan validator such as VAL [11], and a template-based text generator. In the future, we plan to augment explanations with automatically generated metadata about plans, e.g., plan structure [22] and other avenues of using semantics identified by [16] to provide context for richer explanations.

Our ontology-driven approach 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 3, 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 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. Figure 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 [3], 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.

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 .
}
```

This query 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 that illuminate the planner's reasoning at each stage.

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

Table 3: 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 Figure 1.

Type of Explanation	Description	Example Question	Example Response
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.
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.
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".

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. Figure 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 help users understand the trade-offs considered during the planning process.

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
```

This query 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).

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 3). This approach improves AI planning interpretability and advances human-AI collaboration.

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. 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) [17]. This approach can be particularly effective in contexts like LLMs, which have shown promise for automated planning [19]. 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 [23], where iterative prompting strategies are employed, providing feedback of observations from a Plan Validator [11], 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 could offer nuanced insights into coordinating actions and explaining them in a way that is both transparent and informative.

References

[1] Mohannad Babli, Eva Onaindia, and Eliseo Marzal. 2019. Extending planning knowledge using ontologies for goal opportunities. *arXiv preprint arXiv:1904.03606*.

[2] Trevor J. Bihl, Chad Cox, and Timothy Machin. 2019. Towards a taxonomy of planning for autonomous systems. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, 74–79. DOI: 10.1109/NAECON46414.2019.9058324.

[3] Gerard Canal, Senka Krivić, Paul Luff, and Andrew Coles. 2022. Planverb: domain-independent verbalization and summary of task plans. In *Proceedings of the AAAI Conference on Artificial Intelligence* number 9. Vol. 36, 9698–9706.

[4] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2017. Plan explanations as model reconciliation: moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317*.

[5] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256*.

[6] Daniel Garijo and Yolanda Gil. 2012. Augmenting prov with plans in p-plan: scientific processes as linked data. In CEUR Workshop Proceedings.

[7] R Gayathri and V Uma. 2018. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: a survey. *ICT Express*, 4, 2, 69–74.

[8] Malik Ghallab, Dana Nau, and Paolo Traverso. 2004. *Automated Planning: theory and practice*. Elsevier.

[9] Yolanda Gil and Jim Blythe. 2000. 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/*. Vol. 114.

[10] Nicola Guarino, Daniel Oberle, and Steffen Staab. 2009. What is an ontology? *Handbook on ontologies*, 1–17.

[11] Richard Howey, Derek Long, and Maria Fox. 2004. Val: automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 294–301.

[12] Ali Abdul Khaliq, Uwe Köckemann, Federico Pecora, Alessandro Saffiotti, Barbara Bruno, Carmine Tommaso Recchiuto, Antonio Sgorbissa, Ha-Duong Bui, and Nak Young Chong. 2018. Culturally aware planning and execution of robot actions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 326–332.

[13] Benjamin Krarup, Senka Krivic, Daniele Magazzeni, Derek Long, Michael Cashmore, and David E Smith. 2021. Contrastive explanations of plans through model restrictions. *Journal of Artificial Intelligence Research*, 72, 533–612.

[14] Markus Krötzsch, František Simančík, and Ian Horrocks. 2014. Description logics. *IEEE Intelligent Systems*, 29, 12–19, 1.

[15] Timothy Lebo et al. 2013. Prov-o: the prov ontology. *W3C recommendation*, 30.

[16] Freddy Lécué. 2020. On the role of knowledge graphs in explainable ai. *Semantic Web*, 11, 1, 41–51. http://dblp.uni-trier.de/db/journals/semweb/semweb11.html#Lecue20.

[17] Sudip Mittal, Anupam Joshi, and Timothy W. Finin. 2017. Thinking, fast and slow: combining vector spaces and knowledge graphs. *ArXiv*, abs/1708.03310.

[18] Bharath Muppasani, Vishal Pallagani, Biplav Srivastava, and Raghava Mutharaju. 2023. 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)*.

[19] Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Lior Horesh, Biplav Srivastava, Francesco Fabiano, and Andrea Loreggia. 2022. Plansformer: generating symbolic plans using transformers. In *On Arxiv at: https://arxiv.org/abs/2212.08681*.

[20] Stephanie Rosenthal, Sai P Selvaraj, and Manuela M Veloso. 2016. Verbalization: narration of autonomous robot experience. In *IJCAI*. Vol. 16, 862–868.

[21] SPARK. 2024. Scheduling and planning applications workshop. *https://icaps23.icaps-conference.org/program/workshops/spark/, https://ccia.ugr.es/ lcv/SPARK/*.

[22] Biplav Srivastava, Jussi Vanhatalo, and Jana Koehler. 2005. Managing the life cycle of plans. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3* (IAAI'05). AAAI Press, Pittsburgh, Pennsylvania, 1569–1575. ISBN: 157735236x.

[23] Katharina Stein and Alexander Koller. 2023. Autoplanbench:: automatically generating benchmarks for llm planners from pddl. *arXiv preprint arXiv:2311.09830*.

[24] Andre Valente, Thomas Russ, Robert MacGregor, and William Swartout. 1999. Building and (re) using an ontology of air campaign planning. *IEEE Intelligent Systems and their Applications*, 14, 1, 27–36.

[25] Mark D Wilkinson et al. 2016. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3.

[26] Monika Žáková, Petr Křemen, Filip Železnỳ, and Nada Lavrač. 2010. Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering*, 8, 2, 253–264.

[27] Parisa Zehtabi, Alberto Pozanco, Ayala Bolch, Daniel Borrajo, and Sarit Kraus. 2024. Contrastive explanations of centralized multi-agent optimization solutions. In *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 34, 671–679.

A Demonstration of Usecase

A.1 Usecase 1: Identifying Most Promising Planner

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
```

B SPAQRL queries for Competency Question

For the evaluation of the competency questions, we have considered a sample knowledge graph, shown in Figure 4, for blocksworld from IPC-2000 domain created using planning ontology shown in Figure 2.

C1: ***"What are the different types of planners used in automated planning?"*** Identifies and lists the various types of planners utilized in the field of automated planning.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?planner
WHERE {
  ?planner a plan-ontology:planner.
}
```

C2: ***"What is the relevance of a planner in a given problem domain?"*** Explores the importance and applicability of different planners within a specific problem domain.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?domain ?relevance ?planner
WHERE {
    ?domain a plan-ontology:domain;
            rdfs:label "caldera".
    ?planner a plan-ontology:planner.
    ?domain ?relevance ?planner.
}
```

C3: ***"What are the available actions for a given domain?"*** Provides a list of actions that can be performed within a specified domain.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?domain ?action
WHERE {
    ?domain a plan-ontology:domain;
            rdfs:label "caldera".
    ?domain plan-ontology:hasMove ?action.
}
```

C4: ***"What problems in a domain satisfy a given condition?"*** Identifies problems within a domain that meet certain specified conditions.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?problem
WHERE {
    ?domain a plan-ontology:domain;
            rdfs:label "caldera".
    ?problem a plan-ontology:problem.
    ?domain plan-ontology:hasProblem ?problem.
    ?problem plan-ontology:hasGoalState ?condition
        .
    FILTER(?condition = "specified_condition")
}
```

C5: ***"What are all requirements a given domain has?"*** Enumerates all the prerequisites or conditions necessary within a particular domain.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?domain ?requirement
WHERE {
    ?domain a plan-ontology:domain;
```

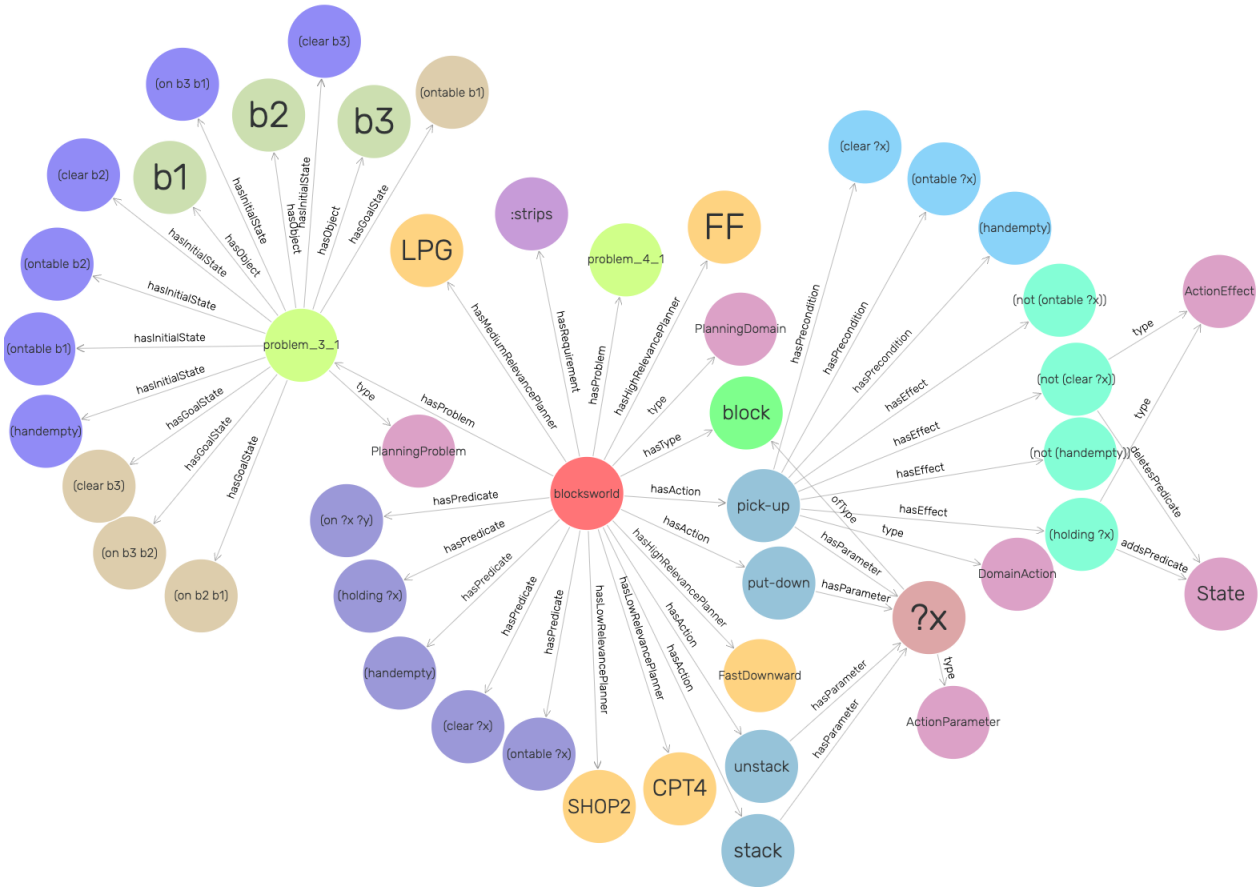

Figure 4: Knowledge graph representation for blocksworld domain from IPC-2000

```
        rdfs:label "caldera".
    ?domain plan-ontology:hasRequirement ?
        requirement.
}
```

C6: ***"What is the cost associated with generating a plan for a given problem?"*** Determines the cost involved in creating a plan to solve a specified problem.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?plan ?cost
WHERE {
    ?problem a plan-ontology:problem;
            rdfs:label "problem-01".
    ?problem plan-ontology:hasPlan ?plan.
    ?plan plan-ontology:hasCost ?cost.
}
```

C7: ***"How many parameters does a specific action have?"*** Counts the number of parameters required for a particular action within a domain.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT (COUNT(?parameter) AS ?parameterCount)
WHERE {
  ?action a plan-ontology:action;
          rdfs:label "get_domain". # action of
              domain caldera
  ?action plan-ontology:hasParameter ?parameter.
}
```

C8: ***"What planning type a specific planner belongs to?"*** Classifies a given planner into its respective planning type.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?planner ?planningType
WHERE {
  ?planner a plan-ontology:planner;
```

```
          rdfs:label "Delfi1".
  ?planner plan-ontology:ofPlannerType ?
      planningType.
}
```

C9: ***"What requirements does a given planner support?"*** Lists the requirements that a specific planner is capable of addressing.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?planner ?requirement
WHERE {
  ?planner a plan-ontology:planner;
          rdfs:label "Delfi1".
  ?planner plan-ontology:solvesRequirement ?
      requirement.
}
```

C10: ***"What are the different parameter types present in a domain?"*** Identifies and lists the various types of parameters present within a specified domain.

```
PREFIX plan-ontology: <https://purl.org/ai4s/
    ontology/planning#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema
    #>

SELECT DISTINCT ?domain ?parameterType
WHERE {
    ?domain a plan-ontology:domain;
            rdfs:label "caldera".
    ?domain plan-ontology:hasParameterType ?
        parameterType.
}
```