Robot Situation and Task Awareness using Large Language Models and Ontologies

Victor Molina* *victor.molina.diez@estudiantat.upc.edu*
Oriol Ruiz-Celada* *oriol.ruiz.celada@upc.edu*
Raul Suarez *raul.suarez@upc.edu*
Jan Rosell *jan.rosell@upc.edu*
Isiah Zaplana *isiah.zaplana@upc.edu*

Institute of Industrial and Control Engineering, *Universitat Politècnica de Catalunya*, Barcelona, Spain

***Abstract*—Robot situation and task awareness requires a deep understanding of the environment, the domain knowledge, and task planning. We present a novel framework that integrates ontologies, Large Language Models (LLMs), and the Planning Domain Definition Language (PDDL) to enhance the comprehension capabilities of robotic systems. The framework employs an LLM to extract structured knowledge from natural language descriptions provided by a human user, populating an OWL ontology that captures relevant objects, properties, and relations. This populated ontology is then used to parse a PDDL Domain file and generate a corresponding PDDL Problem file to solve particular planning problems. This research contributes to the intersection of knowledge representation, natural language processing, and automated planning, providing a solution for intuitive human-robot interaction through LLMs.**

***Index Terms*—robotic manipulation, task planning, ontologies, Large Language Models**

I. Introduction

The advancement of robotic manipulation systems has become a focal point in the development of autonomous and intelligent systems, particularly in dynamic and complex environments. A major challenge in this domain is to enable robots to perform manipulation tasks autonomously while adapting to changes in the environment and responding to various uncertainties. Achieving such levels of autonomy requires the integration of robust planning, execution, and monitoring capabilities, all of which must be dynamically adjusted as new information about the environment becomes available. To this end, knowledge-based frameworks that utilize ontologies [1]–[3] have proven to be promising approaches that facilitate adaptive behaviors.

Ontologies provide the robot with an understanding of the domain, the current situation, and the execution structures necessary to perform the tasks. This includes the ability to reason about the initial and goal states of a task, configure planning and execution processes, and monitor the task in real time. The awareness of the robot about the situation, domain actions, and execution structures enables it to adapt its behavior as needed, ensuring more efficient and flexible task execution in uncertain or dynamic settings. However, even with these capabilities, robotic systems often encounter gaps in knowledge, particularly when sensory data from the environment is incomplete, ambiguous, or misleading. Addressing these gaps is critical for improving the overall autonomy and decision-making capabilities of robotic systems.

To bridge these knowledge gaps and enhance robot adaptability to new or uncertain environments, we propose a novel approach that utilizes human information through Large Language Models (LLMs). LLMs have demonstrated remarkable success in understanding and generating natural language, motivating their integration into the perception and task specification processes of the proposed framework. This integration enables human users to complement the robot's understanding of the environment, enhancing situational awareness and improving task goal specification.

The framework is illustrated in Figure 1. At the center is the Knowledge Database Manager, responsible for updating ontologies with new knowledge and performing ontological reasoning. It uses Owlready2 [4], a Python library that provides efficient access to and modification of ontologies in the OWL (Web Ontology Language) format[1]. Two components populate the ontologies. The Smart Perception Module manages the robot's perception system and constructs an ontology-based world model [5]. This world model captures essential information of the environment, such as object poses and features, and, through the ontology reasoning process, the spatial relations between objects. The second one is the Ontology Populator, the LLM-based component that enables human users to intervene by providing natural language descriptions of the environment. The information provided by human users is used, on the one hand, to complement the current state information (*init*) when the robot's perception system encounters limitations, such as ambiguous object detections or missing data, thereby enhancing the robot's situational awareness. On the other hand, it is used to easily define the desired state (*goal*), complementing task awareness.

Traditionally, robotic tasks are defined through formal programming languages such as the Planning Domain Definition Language (PDDL) [6]. While these languages provide precise representations of tasks, they can be complex and inaccessible to users without technical expertise. In the proposed framework, once the ontology has been populated, a parsing process is employed to map the knowledge contained in the ontology to the predicates defined in the Planning Domain Definition Language (PDDL) domain. After the

* These authors have contributed equally to this work.

[1]https://www.w3.org/OWL/

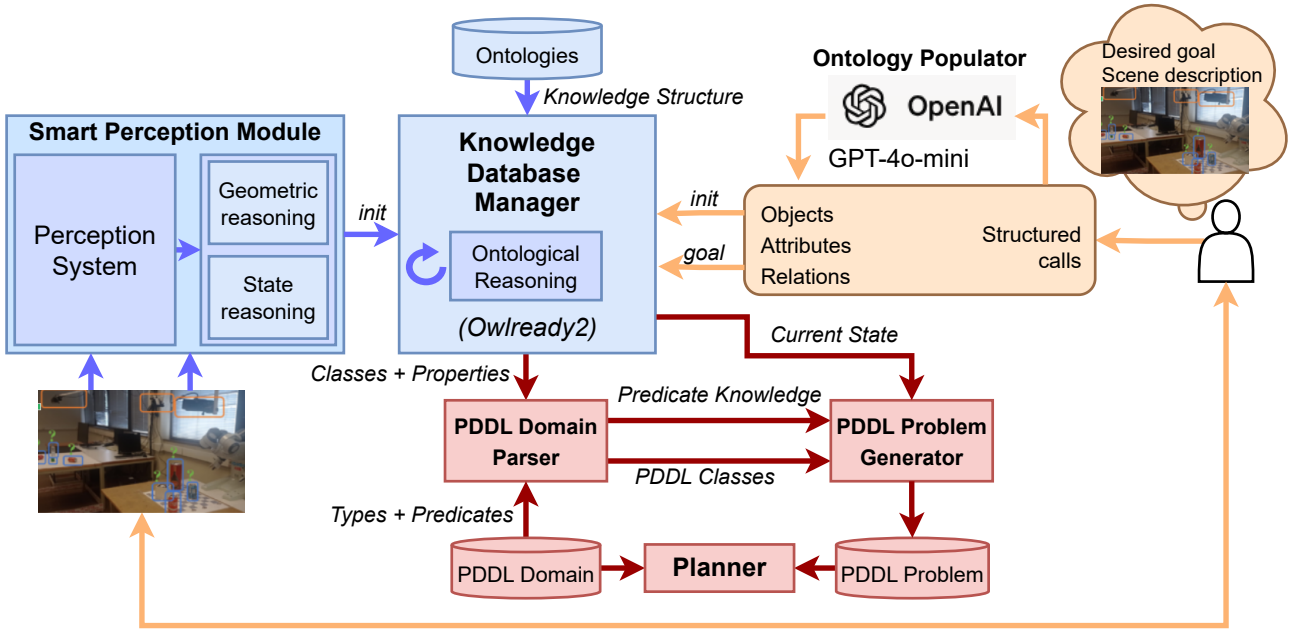


Fig. 1. Proposed knowledge-based framework with LLM ontology population and PDDL generation

parsing, the framework is capable to automatically translate the ontology-based current and desired states of the world into a PDDL problem file, which the robot uses to generate a formal task plan.

The proposal presented here not only enables users to specify using LLMs the initial task goals, but also supports goal updates during task execution. For instance, if an unexpected event occurs that requires the robot to replan or recover from an error, the user can provide a new goal description, and the system will generate the corresponding PDDL representation for the updated task. This dynamic interaction between the user and the robot ensures that the system can adapt to changes in real-time and continue executing tasks even in the presence of uncertainties or partial failures.

The main contributions of the paper are:

- A method to instantiate ontologies using LLMs,
- A method to generate the ontology-based current and desired states of the world into a PDDL problem file.

The paper is structured as follows. Section II presents some related works, Section III introduces the Ontology Populator, Section IV the Ontology-based PDDL generation method and, finally, Section V presents the conclusions of the work.

II. Previous Work

A. Ontology population

Ontology population, the process of instantiating an ontology with domain-specific entities and relationships, is a critical step in knowledge engineering. Traditional methods often rely on manual curation or semi-automated extraction from both structured and unstructured data sources. While manual approaches ensure high accuracy, they are labor-intensive and prone to human bias. Early semi-automated methods, such as those proposed by Kietz et al. [7], Maedche and Staab [8], and Cimiano and Völker [9], leveraged rule-based natural language processing (NLP) techniques but struggled with domain-specific nuances.

Recent advancements in Large Language Models (LLMs) have introduced novel methodologies for ontology population. LLMs, trained on vast text corpora, capture complex linguistic patterns and domain knowledge, thus facilitating the automation of ontology population. For instance, Ciatto et al. proposed using LLMs as oracles for ontology population [10]. Their method utilizes a predefined ontology schema and query templates to automatically generate instances for both classes and properties by iterating over the ontology. Their experimental results demonstrated a significant improvement in accuracy. However, this iterative process can result in higher computational time and increased API calls, as each class in the ontology prompts the LLM to seek entities that match a specific template, such as "examples of ⟨class⟩?".

Similarly, Norouzi et al. explored the use of LLMs for ontology population through prompt engineering techniques [11]. Their approach extracts relational triples from unstructured text using small modular ontologies as guidance, focusing on processing large text data with Retrieval Augmented Generation (RAG), rather than solely on the population of the ontology itself.

Finally, similar to our work, Caufield et al. introduced the Structured Prompt Interrogation and Recursive Extraction

of Semantics (SPIRES) method for populating knowledge bases using zero-shot learning [12]. SPIRES exploits the zero-shot learning capabilities of LLMs to perform general-purpose query answering from flexible prompts, returning information that conforms to a specified schema. Given a detailed, user-defined knowledge schema and input text, SPIRES recursively interrogates prompts against LLMs like GPT-3 to obtain responses matching the provided schema. However, this recursive extraction may lead to increased API calls and computational costs. Moreover, SPIRES primarily focuses on entity and attribute extraction, while our work also aims to extract relations.

B. Generation of Planning files

The generation of task planning files is a well-established field in robotics, as many approaches have focused on creating efficient methods to translate ontological and domain knowledge into formats suitable for task planning algorithms. These approaches enable robots to operate in complex environments by reasoning about tasks, actions, and constraints. While traditional systems often require domain-specific adaptations to translate ontological knowledge into PDDL, recent advancements are moving toward more flexible and dynamic approaches, including the automated generation of planning domains based on observations [13].

The PMK framework [14], for example, employs ontologies to define task sequences, actions, and constraints, enabling robots to adapt dynamically based on real-time perception. This framework is particularly focused on Task and Motion Planning (TAMP) problems, integrating both task-level decision-making and motion feasibility. Instead of using PDDL for planning, PMK queries an ontology with predicates like `robot-reachability-grasping` and `feasible` to evaluate constraints and determine suitable actions. This method trades the broad expressiveness of PDDL for a tighter coupling between task and motion planning, allowing the robot to operate with more immediate feedback on its physical constraints and capabilities. However, this limits the complexity of tasks PMK can handle, as it focuses more on streamlining TAMP interactions rather than general planning.

Kootbally et al. [15] take a more traditional approach to PDDL generation by automating the creation of PDDL problem files specifically for kitting domains. They use an OWL/XML schema to structure the knowledge base and a MySQL database to store and retrieve relevant information about each task. Their method relies on custom-made functions that dictate how predicates are built from the ontology and incorporated into the PDDL file, meaning each domain or task requires specific coding and adaptation to work with the planner. This approach demonstrates the potential for automating task planning in well-defined, repetitive domains but requires significant manual setup for each new application, reducing scalability and flexibility.

Hoebert et al. [3], [16] introduce a more flexible pipeline for PDDL generation by using SPARQL queries to extract relevant predicates from an ontology and integrate them into PDDL problem files. This method is particularly suited for industrial robotics, where tasks are often well-defined and environments are relatively static. The use of SPARQL provides a standardized way to query ontological data, allowing for some adaptability without requiring domain-specific custom functions for every new task. However, since their focus is on industrial applications, where the environment and tasks are predictable, this approach may not be ideal for more dynamic or unstructured domains where greater flexibility is required.

As shown, there is growing interest in the integration of ontological knowledge with planning systems, especially as robots are increasingly deployed in complex and dynamic environments. Previous works have demonstrated the potential of using ontologies to generate PDDL files, but many approaches remain tied to specific domains or require manual adaptation for each new task. Unlike these earlier efforts, this paper introduces a novel PDDL generation mechanism that uses ontologies instantiated through large language models (LLMs) for the generation of the problem files. This approach is domain-agnostic and eliminates the need for custom-built functions to translate ontology individuals and relations into PDDL files. By utilizing LLMs for ontology initialization, the system can handle a wide variety of domains without requiring domain-specific knowledge encoding, offering greater scalability and flexibility for diverse robotic applications.

III. Ontology population with LLMs

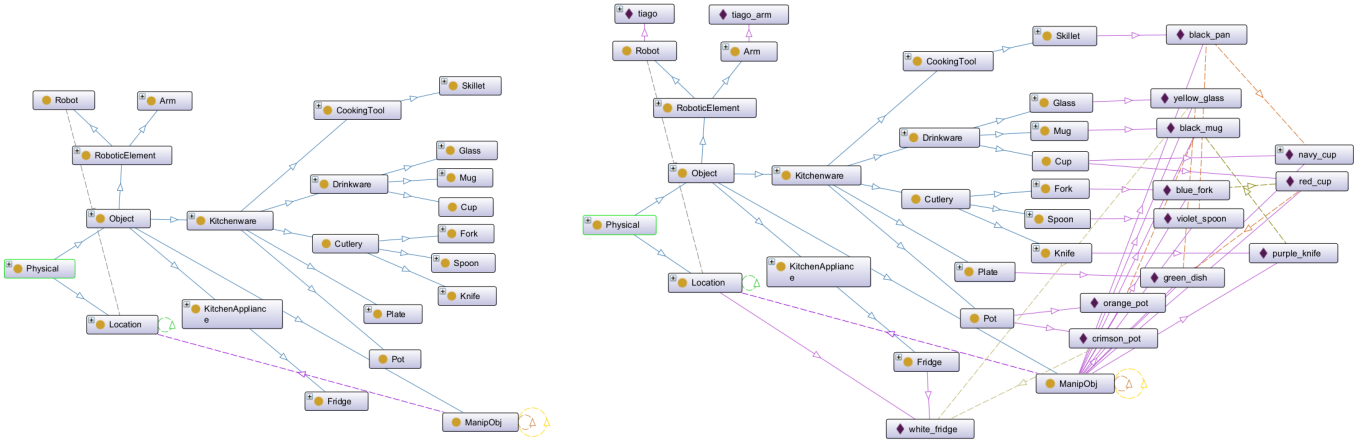
The system developed for the automatic population of ontologies from unstructured text uses the capabilities of Large Language Models (LLMs) in conjunction with domain-specific ontology knowledge. Our system effectively integrates the open-world knowledge provided by LLMs with the more constrained, specific knowledge embedded in the ontology. By doing so, it can accurately identify and map entities, attributes, and relationships mentioned in the text to the corresponding elements defined within the ontology.

The system architecture is designed to take an ontology (Figure 2a) and a corresponding text as input and produce an instantiated ontology as output (Figure 2b), using the Owlready2 library to manipulate and query the ontology in the Knowledge Base Manager.

The LLM selected for this system is the GPT-4o-mini model from OpenAI, which represents one of the state-of-the-art models in natural language processing. OpenAI's models are renowned for their robust capabilities and are fully compatible with the majority of LangChain methods, a framework designed to build applications with language models[2]. Moreover, GPT-4o-mini offers a context window of 128K tokens and can generate up to 16,348 output tokens. This performance makes it an ideal choice for this application.

The population process is organized into a pipeline consisting of three primary steps (depicted in orange in

[2]https://www.langchain.com/



(a) (b)

Fig. 2. **(a) Subset of the ontology before population; (b) Subset of the ontology after population.** The arrows denote different relations. Blue: `subclassOf`; Purple: `hasIndividual`; Orange: `above`; Dark Green: `inside`; Light Green: `in`.

Figure 3): Entity Extraction, Property Extraction, and Relation Extraction. Each step is tailored to handle distinct aspects of the population task by guiding the LLM through a series of prompts (depticted in blue in Figure 3), ensuring that entities, their corresponding attributes, and relationships are correctly identified and instantiated within the ontology framework.

Entity Extraction: The first step in the pipeline focuses on extracting the entities present in the text and mapping them to the corresponding classes in the ontology. This process involves prompt-tuning the LLM through a series of prompts designed to guide the model in identifying and extracting entity mentions that align with the predefined classes in the ontology. These classes are derived directly from the ontology and correspond to the leaf classes, as we aim to instantiate concrete, real-world entities. This approach ensures that the extraction process is grounded in the domain-specific concepts defined within the ontology.

To enhance the accuracy of entity-class matching, the prompt provided to the LLM is enriched with descriptive information for each ontology class. This augmentation helps the model to more effectively align the extracted entities with their corresponding classes, particularly when the class names are ambiguous or lack inherent descriptive clarity. Additionally, it is common that not all entities within the ontology have a complete or precise description, as crafting these can be a labor-intensive task for ontology modelers. To address this challenge, the system utilizes the LLM to autonomously generate descriptions for classes that are not sufficiently detailed. By generating these descriptions, the system augments the class information, ensuring better matching during the entity extraction process. The LLM generates these class descriptions using a specialized prompt designed to capture the class's inherent geometry, function, and relationships with similar entities. This approach is based on the principles suggested by Tang et al. [17], informed by Rosch's theory of cognitive representations of semantic categories, which postulates that categorization is grounded in perceptual and functional similarities among objects. By incorporating these factors into the class descriptions, the system strengthens the alignment between extracted entities and their corresponding ontology classes, even in cases where manual descriptions are absent. This method not only improves the precision of the entity-class matching process but also ensures that entities can be effectively extracted and categorized, regardless of the availability of pre-existing descriptions.

In cases where an entity does not directly match any ontology class (e.g., due to synonym usage or lexical variations), a secondary matching mechanism is triggered. This mechanism uses embedding-based similarity, where embeddings are generated for both the unmatched entity and the descriptions of ontology classes. Then, a cosine similarity measure is applied to identify the closest matching class based on a predefined similarity threshold. For example, in our ontology we have the class `Skillet`, which represents a frying pan. However we do not have the class `Pan` itself. If the human describes a `black_pan` it may happen that the LLM is not capable of recognizing it as a `Skillet`, miss-classifying it as a `Pan` or even not matching it to any class. With our embedding method, the embedding of `black_pan` is extracted and compared with the embeddings of the other classes, resulting with high similarity match with the class `Skillet`, as shown in the example populated ontology (Figure 2b). This approach ensures that only semantically relevant entities are matched with appropriate classes, minimizing the risk of erroneous or random matches.

Property Extraction: The second step in the pipeline involves extracting the attributes associated with each identified entity. Similar to the entity extraction phase, this step also involves prompt-tuning the LLM to extract relevant properties or attributes for each entity. The LLM is instructed to focus on specific attributes that are allowed or expected for

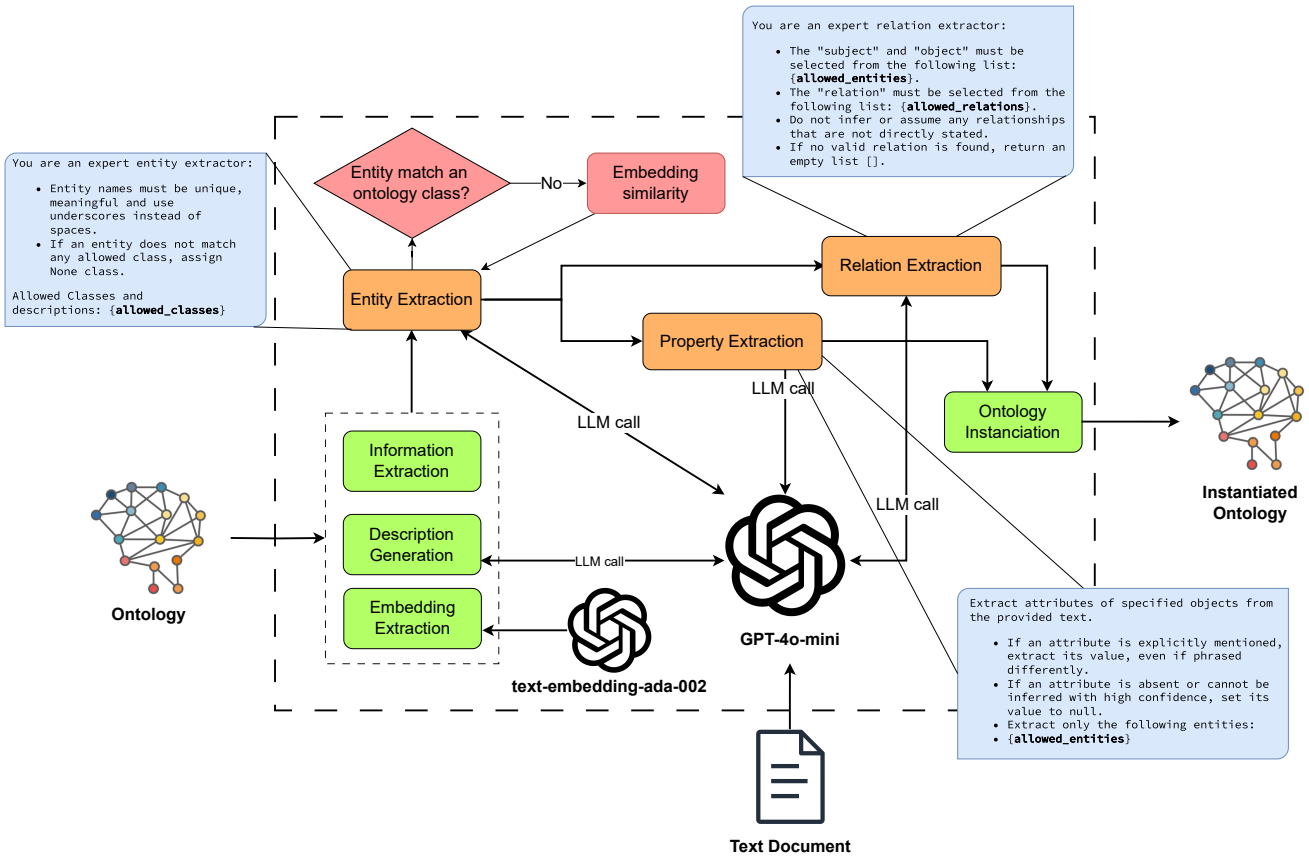


Fig. 3. Ontology population framework

each entity type, based on the attributes range and domain, similar to what the SPIRES framework does [12]. A list of permissible attributes is provided to the model, ensuring that only valid properties are considered. This step ensures that all entities are comprehensively characterized by their associated properties, which are then integrated into the instantiated ontology. The output is subsequently processed to remove any entities that were extracted during this phase but were not previously identified in the entity extraction step, thereby ensuring consistency in the final ontology.

Relation Extraction: The final step in the pipeline is dedicated to the extraction of the relationships between the entities identified in the text. Again, the LLM is prompt-tuned through a series of instructions that guide it to recognize and extract relations. This process ensures that only valid relations, as defined by the ontology, are considered. A list of allowed relations is provided to the model, restricting the possible relationships to those predefined in the ontology. Furthermore, the entities involved in the relations are constrained to those identified in the earlier steps, ensuring consistency and coherence in the resulting ontology structure. The output is further processed to remove any relations where the subject or object are not among the entities identified in the first step or where the relation is not an allowed one, ensuring that only semantically meaningful and valid relations are retained.

It is important to note that in each of the three steps, the LLM is invoked only once per phase to extract all entities, properties, and relations in a single call. This approach avoids the need for iterative calls for each individual class or attribute, which would otherwise increase the number of API calls and computational time. However, this method has its limitations, as LLMs have a fixed context window size, which may be exceeded when processing very large texts with numerous entities and attributes. A promising area for future work involves combining this system with the already discussed Retrieval-Augmented Generation (RAG) approach [11], to handle large-scale documents more effectively.

Ontology population: After the entities, attributes, and relations have been extracted, the final step is the population of the ontology using the Owlready2 library. During this stage, the identified entities, along with their associated attributes and relationships, are incorporated into the ontology. Which is incrementally updated to reflect the newly extracted information.

A key feature of the population process is the dynamic assignment of ontology classes based on the relations. In some cases, the domain or range of a relation may conflict with the class of an entity. For example, consider an entity such

as white_fridge, which is identified as an element of the Fridge class. If a relation like (crimson_pot, in, white_fridge) is extracted, and the relation schema is (ManipObj, in, Location) where the range has to be of type Location (which does not include Fridge as a subclass), the system automatically assigns the Location class to the entity white_fridge to reflect the semantic context of the relation. Same happens if the class Pot for crimson_pot is not a subclass of ManipObj as shown in Figure 2b. This ensures that the instantiated ontology accurately represents both the entities and the logical structure of their interrelationships, similar to what can be seen in the relation phase in [10].

Overall, this system offers an automated approach to ontology population from unstructured text. By leveraging the power of LLMs for semantic extraction, it ensures the consistency and integrity of the ontology while respecting the defined structural constraints and reducing the number of LLM calls.

IV. Ontology-based PDDL Generation

The system developed for the automatic generation of PDDL (Planning Domain Definition Language) problems is based on interpreting and integrating knowledge from the ontology with a predefined PDDL domain file. This process is done in two steps: PDDL Domain parsing, matching the information included in the domain file with the ontological knowledge; and PDDL Problem generation, generating a new problem file based on the current knowledge of the elements in the world.

A. PDDL Domain parsing

The process begins by loading the PDDL domain, which is assumed to be created by a domain expert, to the PDDL Domain parser. The domain defines the set of predicates, actions, and types outlining the planning problem space. The information in the domain file is parsed in the following way:

- **Types:** Types define categories of objects in the domain, helping to organize them. For instance, in a robotics context, types could include Robot, ManipObj, or Location. The parser performs a direct match between the types in the domain and classes in the ontology. Using the reasoning capabilities of the ontology, individuals can be classified into these classes even when not explicitly asserted to belong to them by the user, the perception module, or the LLM ontology instantiator. These classes are identified as **PDDL Classes**.
- **Predicates:** Predicates describe relationships or properties of these objects, representing the state of the environment. For example, a predicate like (holding ?robot ?object) indicates that a robot is holding an object. Predicates are used to define the preconditions and effects of actions, guiding the planner in achieving goals. Following the previous example, the predicate (holding ?robot ?object) is a precondition of the action (place ?robot ?object ?location). Predicates indicate the relationship between elements in the domain or the state of an element. The parser is employed to extract the predicates defined in the PDDL domain and matching them with the corresponding property in the ontology. However, not all the properties in the ontology will have a direct match with a predicate from the PDDL domain. Even in cases where there is no explicit match, these additional ontological properties may still contribute to reasoning about the state of the world and other properties that do belong to the predicate space. Consequently, the system defines a subset of the properties of the ontology that are directly relevant to the PDDL domain as **Predicate Knowledge**. This subset of knowledge is essential for populating the predicates in the initial state and goal of the generated PDDL problem.

The parsing of the domain is performed at initialization, as the domain is constant. If there is a mismatch between the domain file and the ontology, such as a predicate not having any matching ontological property, a warning is sent to the user, ensuring consistency between PDDL and ontological knowledge.

B. PDDL Problem generation

Once the predicates and types have been matched with corresponding properties and classes in the ontology, the system is ready to generate a PDDL problem instance for any specific scenario. New instances and their properties, which may be updated by the perception module or generated by an LLM-based ontology instantiator, are accessed via the Owlready2 library. This allows for programmatic exploration of the content of the ontology, ensuring that the most up-to-date knowledge is utilized in problem generation.

When a new goal is received, the system has to match the parsed PDDL Classes and Predicate Knowledge to a new PDDL Problem file. By iterating over the PDDL Classes, the generator identifies all relevant instances (i.e., individuals) that belong to these PDDL types. These instances, representing objects in the environment, are considered for inclusion in the planning problem, and are listed in the :objects field in the PDDL Problem.

To obtain the initial state of the PDDL Problem in the :init field, the system iterates over the instances corresponding to the PDDL Classes. For each instance, it examines the properties that belong to the Predicate Knowledge subset of the ontology. Ontological properties are classified into two categories: object properties and data properties.

- **Object properties:** Object properties describe relationships between two individuals and can be directly translated into PDDL predicates in the form (predicate ?subject ?object). For example, an object property such as isOnTop would be translated into the predicate (isOnTop can1 table1) in the PDDL problem, where can1 and table1 are specific indi-

viduals in the ontology belonging to the PDDL Classes `ManipObj` and `Location` respectively.

- **Data Properties:** Data properties, in contrast, describe attributes of a single individual and may have various data types. For PDDL problem generation, the system focuses on boolean data properties that correspond to the predicates defined in the domain. The value of these boolean properties determines whether a predicate is included in the initial state of the PDDL problem. If the boolean property is `True`, the corresponding predicate is added to the initial state. If the property is `False`, the predicate is omitted. For example, the boolean property `gripperEmpty` associated with an individual `robot1_gripper` would be translated into the predicate `(gripperEmpty robot1_gripper)` if the value of the property is true, indicating that the gripper of `robot1` is empty.

The system uses the Pellet reasoner [18], an OWL-DL (Web Ontology Language – Description Logic) reasoner, to infer additional relationships and classifications based on the logical axioms defined within the ontology. This reasoning process enables dynamic classification of individuals by inferring properties and relationships not explicitly stated. For example, Pellet may infer that an individual belongs to a particular class or satisfies specific conditions, allowing that individual to be included or excluded from the planning domain. By leveraging Pellet's reasoning capabilities, the system ensures that the PDDL generation process is based on a logically consistent and complete set of knowledge, extending beyond what is directly stated in the ontology.

An example of a Domain and the generated Problem PDDL files based on Figures 2 (a) and (b) is shown in Figure 4. Once the PDDL Problem has been generated, it is formatted and passed to a planner for execution. In this case, the system integrates the Fast Forward (FF) planner, a well-established heuristic search-based planning framework [19]. The PDDL Problem file generated by the system is input into FF along with the corresponding PDDL Domain file. The planner then uses this information to compute a solution plan that satisfies the initial conditions and achieves the desired goals, based on the actions and predicates defined in the domain. This integration enables the system to automatically generate plans for complex domains, leveraging the dynamic knowledge base provided by the ontology and the inferential capabilities of Pellet. Through this approach, the system can generate solutions in evolving environments where knowledge is continuously updated and expanded through reasoning.

This method, which integrates ontological reasoning with the structure of a PDDL Domain, facilitates the dynamic generation of PDDL Problem instances. By incorporating inferential reasoning from the ontology, the system ensures that only relevant instances and their properties are included in the problem, based on both explicit domain definitions and inferred knowledge. This approach expands the capabilities of traditional methods by enabling automatic and flexible inclusion of individuals in the problem space, depending on their inferred classification through the ontology. Consequently, the system is capable of dynamically updating the planning problem, reflecting changes in the knowledge base, and enabling a more adaptive approach to problem generation, particularly in domains where knowledge evolves over time or new information is continuously integrated.

```
(:types Robot Location ManipObj Arm)
(:predicates
...
(inside ?obj1 ?obj2
(in ?obj ?loc)
(above ?obj1 ?obj2)
(gripperEmpty ?arm)
...)
```

```
(:objects
black_pan yellow_glass black_mug
    navy_cup red_cup blue_fork
    orange_pot crimson_pot violet_spoon
    purple_knife green_dish ... -
    ManipObj
tiago - Robot
tiago_arm - Arm
white_fridge ... - Location)

(:init
...
(inside blue_fork red_cup)
(in crimson_pot white_fridge)
(in yellow_glass white_fridge)
(above green_dish yellow_glass)
(gripperEmpty tiago_arm)
...
)
```

Fig. 4. Top: Part of the PDDL Domain file for a kitchen scenario, showing predicates matching the properties shown in Figure 2 (a). Bottom: Part of the PDDL Problem file generated based on the populated ontology in Figure 2 (b).

V. Conclusions and Future Work

By incorporating the LLMs open-world knowledge capabilities with the domain-specific knowledge provided by the ontology, our ontology population pipeline is capable of populating a given ontology with all the entities, attributes and relations extracted from an unstructured descriptive text provided by the user, enhancing the human-robot interaction and automatizing the labor-intensive and prone to human bias work of populating an ontology manually. Moreover, our population framework is able to reduce the number of calls passed into the LLM, avoiding multiple API calls and iterating over all the ontology classes to find a match, while adding automatically generated descriptions to enhance the context

and an embedding similarity system to mitigate empty matches or possible LLM hallucinations into classes not contained in the ontology.

Furthermore, by integrating ontological reasoning with traditional PDDL domain structures, the proposed system enables the dynamic generation of PDDL problem instances that reflect the evolving state of knowledge represented within the ontology. This system expands upon previous approaches by incorporating inferential reasoning capabilities from ontologies, which allows for more flexible and intelligent planning problem generation. As a result, the system can automatically adjust the individuals and their properties included in the planning problem, ensuring that only relevant entities and conditions are considered based on both explicit definitions and inferred knowledge. This capability is especially useful in complex domains where knowledge evolves over time or where the planning domain needs to be dynamically updated based on new perceptions or information.

As future work, the ontology population system could be improved in order to handle large descriptive texts that surpass the window of context of the LLM, even though the implementation in our particular framework for Robot Task Awareness does not require it, as the user will not write texts of that lengths. Moreover, other applications of LLMs for the knowledge-driven robotic manipulation will be explored, by integrating the LLM for querying over the ontology from natural text or automatically generating a problem file from the ontology knowledge and a human task description.

ACKNOWLEDGMENT

This work was supported by the European Commission's Horizon Europe Framework Programme with the project Resilient manufacturing lines based on smart handling systems (SMARTHANDLE) under Grant Agreement 101091792.

REFERENCES

[1] O. Ruiz-Celada, A. Dalmases, R. Suárez, and J. Rosell, "BE-AWARE: an ontology-based adaptive robotic manipulation framework," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–4.

[2] M. Beetz, D. Bessler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, and G. Bartels, "KnowRob 2.0 - A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 512–519, 9 2018.

[3] T. Höbert, W. Lepuschitz, M. Vincze, and M. Merdan, "Knowledge-driven framework for industrial robotic systems," *Journal of Intelligent Manufacturing*, vol. 34, 3 2021.

[4] J.-B. Lamy, "Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies," *Artificial Intelligence in Medicine*, vol. 80, pp. 11–28, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0933365717300271

[5] O. Ruiz-Celada, A. Dalmases, I. Zaplana, and J. Rosell, "Smart perception for situation awareness in robotic manipulation tasks," *IEEE Access*, vol. 12, pp. 53 974–53 985, 2024.

[6] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, "PDDL - The Planning Domain Definition Language," 08 1998.

[7] J. Kietz *et al.*, "Kaon: A framework for extracting ontology instances from text using rule-based nlp," *Journal of Natural Language Processing*, vol. 12, no. 3, pp. 234–249, 2000, introduced the "KAON" framework, utilizing rule-based NLP to extract ontology instances from text, but faced difficulties with domain-specific language and entity recognition.

[8] A. Maedche and S. Staab, "Text-to-onto: A system for ontology learning from domain-specific corpora," in *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management*, Paris, France, 2001, pp. 246–257, developed the "Text-To-Onto" system, which employed rule-based methods but faced limitations in handling specialized terminologies.

[9] P. Cimiano and J. Völker, "Ontolearn: A hybrid approach to ontology learning from text," *Journal of Artificial Intelligence Research*, vol. 22, no. 4, pp. 81–112, 2005, proposed the "OntoLearn" system, combining rule-based NLP and machine learning for ontology learning, but struggled with domain-specific expressions.

[10] G. Ciatto, A. Agiollo, M. Magnini, and A. Omicini, "Large language models as oracles for instantiating ontologies with domain-specific knowledge," 2024. [Online]. Available: https://arxiv.org/abs/2404.04108

[11] S. S. Norouzi, A. Barua, A. Christou, N. Gautam, A. Eells, P. Hitzler, and C. Shimizu, "Ontology population using llms," 2024. [Online]. Available: https://arxiv.org/abs/2411.01612

[12] J. H. Caufield, H. Hegde, V. Emonet, N. L. Harris, M. P. Joachimiak, N. Matentzoglu, H. Kim, S. A. T. Moxon, J. T. Reese, M. A. Haendel, P. N. Robinson, and C. J. Mungall, "Structured prompt interrogation and recursive extraction of semantics (spires): A method for populating knowledge bases using zero-shot learning," *Bioinformatics*, vol. 40, no. 3, p. btae104, 2024. [Online]. Available: https://doi.org/10.1093/bioinformatics/btae104

[13] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," 2021. [Online]. Available: https://arxiv.org/abs/2105.13604

[14] M. Diab, A. Akbari, M. U. Din, and J. Rosell, "PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation," *Sensors 2019, Vol. 19, Page 1166*, vol. 19, p. 1166, 3 2019.

[15] Z. Kootbally, C. Schlenoff, C. Lawler, T. Kramer, and S. Gupta, "Towards robust assembly with knowledge representation for the planning domain definition language (pddl)," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 42–55, 2015, special Issue on Knowledge Driven Robotics and Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584514000672

[16] Hoebert, Timon, Lepuschitz, Wilfried, and Merdan, Munir, "Automatic ontology-based plan generation for an industrial robotics system," 2020.

[17] C. Tang, D. Huang, W. Ge, W. Liu, and H. Zhang, "Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 2023–2030, 2023. [Online]. Available: https://doi.org/10.1109/LRA.2023.3297896

[18] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007, software Engineering and the Semantic Web. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570826807000169

[19] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, pp. 253–302, 2001.