**TALLINN UNIVERSITY OF TECHNOLOGY**

SCHOOL OF ENGINEERING

Department of Electrical Power Engineering and Mechatronics

# KNOWLEDGE REPRESENTATION AND PROCESSING FOR ROBOTS

EEM0050 REPORT

Student: Kayode Hadilou ADJE

Student Code: 194360MAHM

Supervisor: Sashidhar Kanuboddi, Research Assistant, Institute for Artificial Intelligence, RWU, Germany

Weingarten, 2021

**Table Of Contents**

**List of Figures**

# 1. OVERVIEW

## 1.1 Host Organization

The host organization is the Institute for Artificial Intelligence (IKI in German) at Ravensburg-Weingarten University of Applied Science in Weingarten, Germany. The institute's main task is to perform research on ways to make machines, robots more adaptive and intelligent as humans. This institute is member of the Baden-Württemberg Center for Applied Research in the category of technology for intelligence systems. The institute is led by Prof. Dr. Wolfgang Ertel who has more than 20 years in machine learning research in many areas including robotics, medicine, education, etc. The other members of the institute are mainly research assistants and associate who work on different projects involving partnerships with either private companies or public institutions. One of the project the institute is currently working on is Robotkoop [1]: it consists of developing intelligent algorithms which help the robot mimics humans intelligence. The research robot Tiago [2] from Pal Robotics is used, the project is funded by the German Federal Minister of Education and Research.

## 1.2 Task Description

The task given to me at IKI, the host organization consists of researching and implementing knowledge processing and representation methods for service robots. KnowRob [3] is the most daring and advanced framework currently available for knowledge processing in robotics that has been developed for the last 10 years, the first task consisted on understanding KnowRob, its different components and how to implement it for use cases within Robotkoop project. Currently, Knowrob is under maintenance: the core ontology used has been changed to allow usage of newer ontology standard DUL [4] accepted and used in other fields, activity-related concepts are now based on SOMA ontology [5] making the system publicly available on the knowRob's github account unusable for now. So, after a thorough investigation of the project, a software package using Python3 binding is to be proposed to represent and process knowledge for the robot, the workspace environment and a manipulation task. The package is to be delivered in a docker container together with instructions on how it works.

The given task gives an industrial knowledge on knowledge processing based on ontology specially in robotics, it also helps refresh up basics' robotics programming skills based on ROS stack. The task leverages modern software development methodologies including version control and containerization.

# 2. INTRODUCTION

Knowledge representation is a general field of artificial intelligence that deals with how information can be represented so that it is helpful in solving complex tasks. In robotics, knowledge representation and processing has gained more importance due to the increased complexity of robotic systems and tasks robots are required to perform autonomously. For example, domestic service robots are required to autonomously know how to prepare dinner table, how to take care of elderly people, etc. It is almost impossible to explicitly define all actions, sub-actions, the interaction between the robot and its environment, etc. required to perform such tasks, hence the need for knowledge-based approaches.

Ontologies are a perfect example of how information can be represented to allow the extraction of semantic knowledge usable in robotics perception and manipulation tasks. Ontologies provide rich semantic description of robots, their environments, and tasks; by reasoning on ontologies robots can access deep semantic and well-modelled information on complex manipulation tasks with higher level of details than black box artificial intelligence methods.

This project's report first describes KnowRob [6] which is the most advanced and daring research project on ontology-based knowledge processing for robots, then followed the description of an ontology-based method software package for manipulation tasks.

# 3. RELATED WORK

## 3.1 Ontologies

On ontology is "an explicit, formal specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased incomplexity" [7]. Given a domain of interest, an ontology helps in understanding the naming, categories, entities, properties, relations between classes that apply to this domain. Ontologies serve to adopt a common representation of the domain of interest and organize data into knowledge to be used for solving tasks and problems.

Ontologies can be represented as either first order logic, description logic DL or expressive logic. DLs are of particular interest in robotics as they provide reasons and formalism about given concepts in the ontology. The Web Ontology Language or OWL is an ontology representation language for semantic web that uses description logic. OWL is an extension of Resource Description Framework RDF, RDF Schema or RDFS which are all standards of World Wide Web W3. RDF introduces only direct relationships between objects; in RDF it is impossible to map object-oriented relations such as inheritance or subclass. RDFS or RDF Schema fixes this by extending RDF with object-oriented properties such as inheritance, multiclass. But the fact that RDFS was based on RDF still causes an issue and makes the syntax of RDFS complicated. OWL is the last of the list and have been built with an object-oriented design in mind. It supports object-oriented features included in RDFS and even more such as instances, disjointment, or even more description logics such as union, equivalence, intersection which were not directly doable in RDFS.

One thing is to have ontologies, the other is to be able to run queries and get some answers. The W3 standard for querying semantic web is SPARQL. It supports queries on RDF-like databases. Wrappers exist in many other programing languages such as PySPARQL. It is based on SQL and adds some semantic meaning to queries. However, It does not support full OWL. For OWL ontologies, one alternative is to serialize the ontology to obtain N triples (subject + predicate + object) then use SPARQL to perform queries.

Programming interfaces allow both ontology modification and query. If query languages can be seen as data first, programming interfaces can be seen as code first approaches. RDFLIB is a library available to manipulate RDF ontologies in programming languages such as Python which supports SPARQL with its own SPARK implementation.

OWLready[8] supports RDF, RDFS, and OWL ontologies fully. It is possible to wrap an ontology to a python object and manipulate it as a normal python object. it is simpler, uses dot annotation (object-oriented programming). For instance, one can access a subclass using *superclass.class* annotation. It also supports simple queries in normal python and

more complex queries with SPARQL. It also supports reasoner (HELMIT and PELLET), reasoners are used for automatic classification of axioms. Let's say, some new modifications on the ontology, this may have affected (added or removed) some axioms indirectly in the knowledge base, reasoners ensure that the new modifications' results are included in the knowledge base.

## 3.2 KnowRob

KnowRob [6] is an end-to-end framework built at the University of Bremen for knowledge representation and reasoning in robotics. It can be divided in 3 layers: API or interface layer, logic-based programming and core or hybrid layer. Below, we describe each layer and provide information on how they are corelated using a down-up approach.

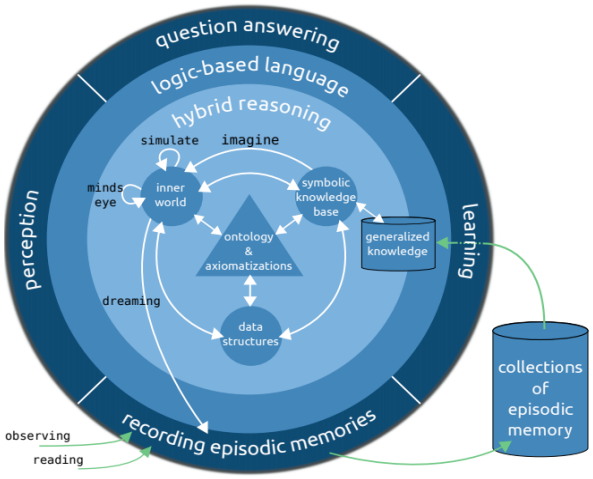



Figure 3.1 KnowRob 2.0 Knowledge processing framework for cognition-enabled robotics agents [6]

### 3.2.1 Hybrid reasoning

At the lowest level of reasoning, we have the following knowledge bases or inference mechanisms: The inner world knowledge base which comes from the reconstruction of the world added to physics simulation; the virtual knowledge base computed on demand from the data structures of the control system; logic knowledge base where sensor and actions data are combined with symbolism (axioms and inference mechanisms); episodic memory knowledge base which are information coming from past experiences of the robotic agent. Above-mentioned knowledge bases can be inconsistent or redundant, KnowRob then uses

a hybrid approach to compute and validate multiple hypotheses' plausibility and consistency.

### 3.2.2 Inner world

KnowRob reconstructs the robot world virtually and add some physics simulation and sensing capabilities- this virtual environment is called the inner world. Objects in the environment are modelled with their CAD and meshes and symbolic annotations. Axioms are used to describe relations between objects and robots.

For example, the following is how a cupboard is represented in the inner world: a cupboard consists of a door with handle, a shelve and an articulation model that the robot uses to open the cupboard. When pushed, a box on a cupboard would tip over an can fall down.

The inner world is necessary because it allow the robotic agent to oversee in its own mind (simulated environment) consequences of its actions during manipulation tasks and therefore adjust the actions it should perform in real environment: this is referred to as simulation-based reasoning and helps avoids the frame problem. For example, the robot can imagine if adding an object to the fridge will cause a tilt, or if displacing its arms will make a cup on a table fall, etc.

### 3.2.3 Narrative-enabled Episodic Memories or NEEM

Episodic memory is the mechanism that allows humans to remember special experiences. As humans, it is easy for us to replay an episode of a movie with detailed information. KnowRob integrates episodic memory through what is called narrative-enabled episodic memory or NEEM. These are video recordings of actions/activities performed by a robot with a detailed story about the actions, motions, their purposes, the effects, the behavior generated, the images captured, the outcomes etc. In the term NEEM, Experience refers to video recording of activities while Narrative refers to a detailed story of what happens.

Consider that the robot wants to clear a dinner table- the activity here is clearing the dinner table. The NEEM will consist of a video of the task plus a detailed narrative as follows: the robot first performs a perception task which is a model with its semantic meaning, then captures the image and focuses on a region of the image and the pose computed. One example of assertion that can be made from this task is that an event called a clean dinner table happened between t1 and t2 times and implies the left arm actuator and an instance of the pose object with values equals to x1, y1, z1.

NEEMs have two main advantages: the robotic agent can perform a query about how a task/ an activity happened. For example, how did I pick up a cup? How did I clean the

dinner table, which body part did I use? Etc..; the collection of NEEM can be used to infer general knowledge about tasks/activities using machine learning methods. In fact, KnowRob offers an interface to the Weka ML (Machine Learning) framework and thus to many learning algorithms.

### 3.2.4 Logic-based language

As it can be seen at the lowest layer of the framework in Figure 3.1, inference mechanisms provided by the hybrid reasoning are very heterogeneous and accomplishing some complex tasks may require more than one inference mechanism. Also, with the representation of knowledge with unstructured large sets of relations or predicates, there is a lack of modularity which makes knowledge engineering difficult. Logic-based language facilitates the use of heterogeneous reasoning mechanisms by providing a unique interface to the reasoning kernel. The interface follows an object-oriented approach where everything is represented as entities which are also retrievable by providing certain metadata about them. Manipulation tasks are described as events during which states of entities change.

### 3.2.5 Highest level interface: querying, perception, recording, learning

This part is the API available to communicate with the rest of the robot control code. The query answering uses KnowRob Querying Language, a logic language built on top of Prolog to access knowledge from hybrid reasoning using query and results. Recording enables the robot to record his experiences as episodic memories and Learning enables the robot to infer general knowledge from the database of collections of episodic memories using Weka ML Framework. At the perception module lies the perception system that can be extended according to domain application. The perception implies that sensorics data is used to help the robotic agent comprehend its surrounding.

Even though KnowRob is a decade year old project described in many articles, the main implementation on GitHub is old and currently being updated. The documentation and tutorials are outdated and unusable at their current state. The ontologies languages and references are being changed for better reorganization and compatibility to other knowledge-based systems.

# 4. ONTOLOGY-BASED KNOWLEDGE PROCESSING FOR MANIPULATION TASKS

This section introduces a package developed during the project that deal with ontology-based knowledge representation and reasoning for manipulation tasks. The package is inspired by how some features of KnowRob and is developed to serve as reference for further improvements. The package consists of an ontology based on OWL for modelling the robot, its environment, and tasks; a spatial reasoning module to enable the robot to geometrically reason about objects around him and task requirement module that adds some tasks related knowledge to the ontology.

## 4.1 Ontology

The ontology described in KnowRob is used as reference to develop an ontology for modelling the robot, its environment and some manipulation tasks. The ontology editor Protégé is used to view and edit, OWLReady is also used with Python3 to provide a binding between the ontology and the robot control code. The ontology is divided into classes, individuals, properties, relations, instances.

Classes are used to represent objects, individuals, or instances. Body parts of the robots are represented as different classes all inheriting from a super class *BodyPart* which is also a subclass of *owl.thing.* The robot and different objects in the scene are each represented by their own classes. Instances are for example instances of classes such as *left_arm* and *right_arm* (instances of the class *arm*). Each class can have relationships with other classes or have simple properties. Classes can have data properties, for example the payload of a robotic arm can be a data property attached to the *arm* class. The speed is attached to the mobile base class and is data property that define the maximum speed of the base of the robot. The (parallel) grippers of a robot have fixed maximum distance they can reach, defined as maximum reachable value. Subclasses of a classes automatically inherit data and object properties of the parent class.

The robot (Tiago) is also defined as a class which has all subclasses of *BodyPart* as object properties. The robot can hold a position *p* in space, the position is an object made of the position and orientation values of the robot.

The class diagram describing classes in the ontology is given in Appendices.

Table 4.1 Object and data properties in the ontologies and their descriptions

| **Object Properties** | |
|---|---|
| *has_dimension* (height, length, and width) | A dimension 3D object containing the height, width and length of a class. |
| *hold_pose* (x, y, z) | A 3D point pose object with value corresponding to pose on x,y and z axis. |
| *has_surface* | A 4d rectangular up-right surface object with the first two values corresponding to the top-left point's x and y coordinate, the 3rd value to the width and the 4th value to the length. |
| *has_part* | A class which is a subclass of *BodyPart* is part of another class such as robot |
| *is_part_of* | A class subclass of BodyPart is part of another part. This object property is an inverse of *has_part.* |
| *has_base_link* | An object's base link according the robot description file |
| *has_end_link* | An object's end link based on robot description file. |
| **Data properties** | |
| *has_grasp_region* | It is a functional property containing that evaluates to true if the object has a grasp region i.e is graspable. |
| *has_max_speed* | It is a functional property containing a double value referring to the maximum speed of a robotic base. |
| *has_payload* | It is a functional property containing a double value referring to the payload of a robotic arm. |

| | |
|---|---|
| *has_reach* | It is a functional property containing a double value referring to the maximum reach of a robotic (parallel) finger. |
| *has_lift* | It is a functional property containing a double value referring to the maximum lifting capability of a robotic torso. |
| *has_spatial_location* | The spatial location of an object with regards to another object |

## 4.2 Spatial Reasoning

It is also common to provide the robotic agent with a situational analysis of its environment by means of geometric reasoning. This consists of relating the objects in the knowledge base and reason over them. In the proposed software package, a module is dedicated to spatial reasoning. Object can have a spatial location property which relate them to other objects in the world [9] . The four spatial reasoning mechanisms used are introduced as followed:

- **right*:*** An object A is said to be on the right of another object B is the difference of coordinates on the X axis of both objects is less than a predefined value of 30cm; and the object A has a higher coordinate on Y axis.
- **left*:*** An object A is said to be on the left of another object B if the difference on Y coordinates is less than 30 cm and the object A has the smallest coordinate on Y axis.
- **inside:** An object A is said to be inside another object B, if the object A can fit in the object B on its width, depth and length and there will still be at least a free space of length equals to a predefined offset of 20cm.
- **on:** An object A is said to be on top of another object B, if the coordinate on z axis of the object A is higher than the coordinate of object B on z axis and the difference of coordinate is negligeable (less than 10cm).

Every object in the ontology can have a spatial location context which relate it to other objects, this property is given as *object1 spatial context object2*, e.g. *Can_1 top table_1*. Figure 4.1 shows all the possibilities of spatial reasoning over objects in the ontology.

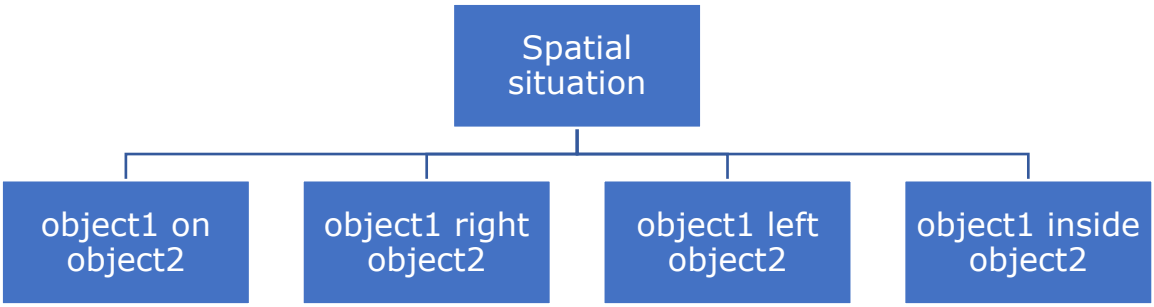



Figure 4.1 Spatial Reasoning on objects in the ontology

## 4.3 Task Planning and Constraints

Modelling tasks and actions is more complex than describing the semantics of a robot as it requires explicit knowledge of the task to be performed and consideration of tiniest details so that deep semantics knowledge can be extracted can be extracted from the task as much as possible. A good example of task description is introduced in DUL ontology [4] which provides an explicit definition and representation of tasks, events and actions. In this software package, DUL is used as reference to defines simpler classes to model manipulation tasks.

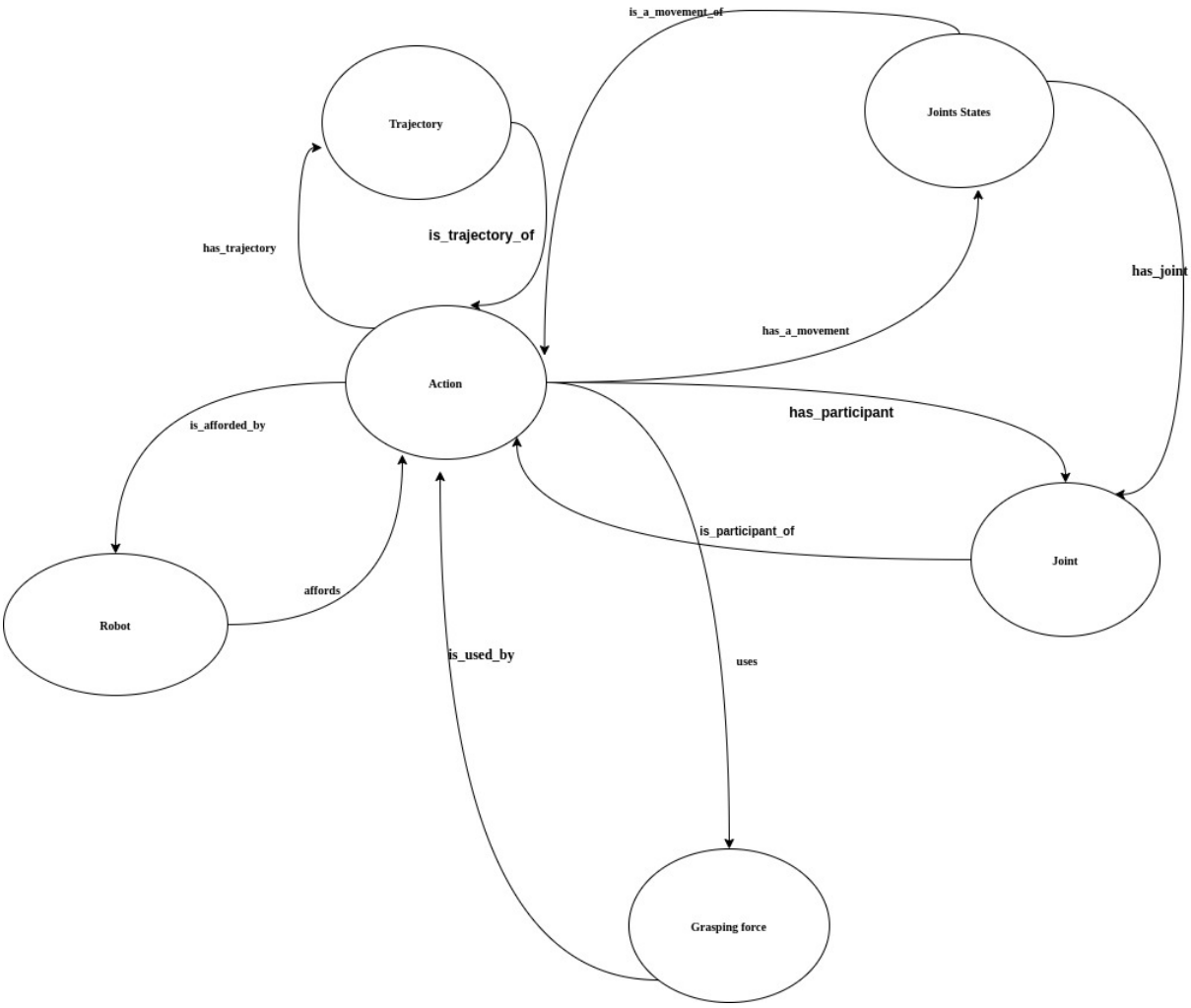



Figure 4.2 Task planning - Class diagram. The figure shows that an action or task (pick up and place is used as reference) can be afforded by the robot based on the definition of affordance described later in this section, an action can have a trajectory, a grasping force can have body parts of the robots or joints as participants. The changes in joints' states during the action is also recorded and saved.

Apart from the classes and relationships described in above figure, we also introduce the concept of affordance and availability. The affordance is the ability of an agent to be able to perform an action i.e., can the arm of a robot successfully grasp an object? In the proposed package affordance, reachability and lifting ability are tightly coupled together: the robot can afford any grasping task if its arms can reach the goal object and the mass of the goal object is less than the payload of the robot (the maximum lift capability of Tiago for example). Furthermore, a region is said to be available if it contains nothing: there are no object on top of the region. This last option can be used to decide where to place an object for example.

## 4.4 Study Case

To showcase the capability of the proposed package, we will use Tiago [2] from Pal Robotics in simulation mode with Gazebo. The picture below shows the simulation scene .

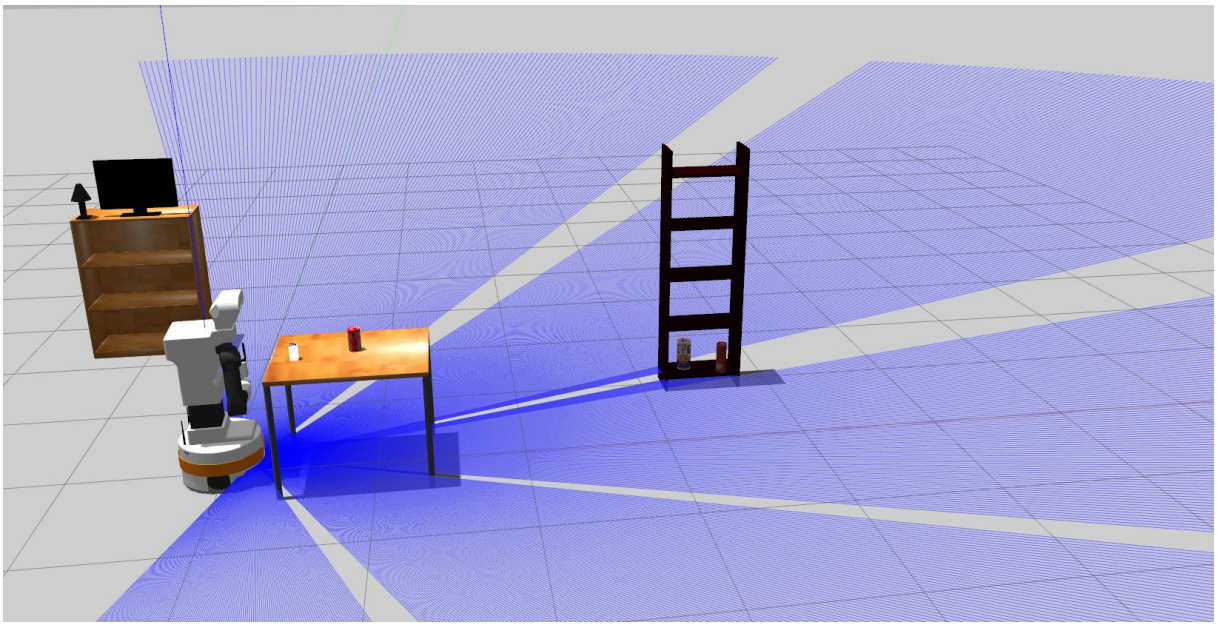


Figure 3 Study Case Simulation Scene

The scene contains the robots, some objects with different shapes and poses to highlight the spatial reasoning capability. Using the scene, the robot will be asked to perform a pickup and place task where the goal object is the white cube on the table in front of the robot. When the robot is performing the task, data about the robot and the task will be recorded and represented in the knowledge base, later on, this knowledge can be viewed in Protégé [10] desktop or web application , queried using python binding with OwlReady, description logic (DL) or SparkQL.

After populating the knowledge base using previously introduced ontological representation, one can visualize the inferred ontology in Protégé.

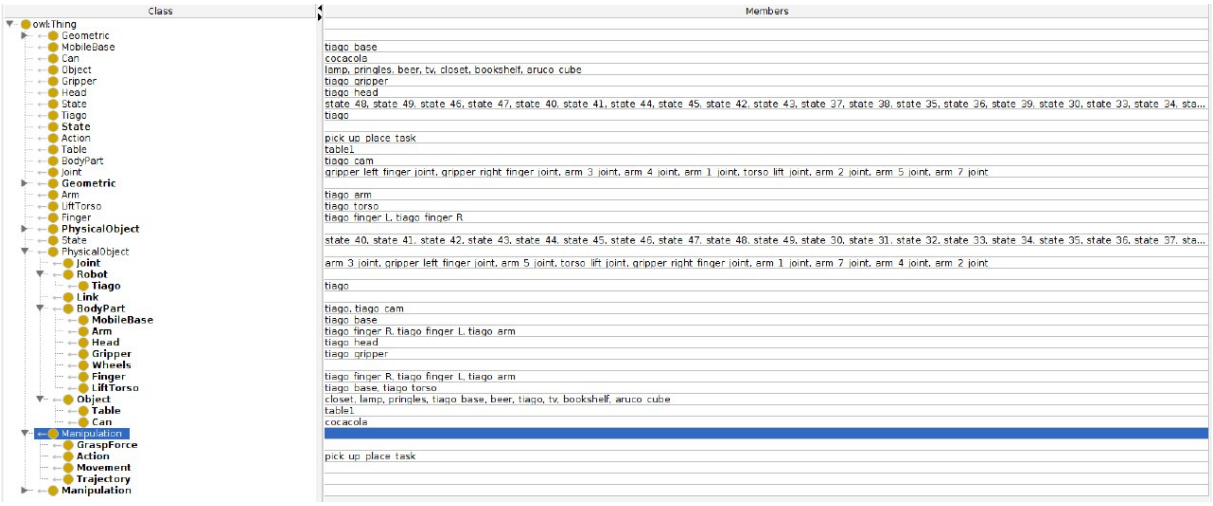



Figure 4 Inferred Ontology's class matrix. In the class matrix, members or instances can be seen together with their instance types. For example, a representation of a robot named *tiago,* some instances of body parts and joints of the robots with their class type, the objects in the work scene and some task's related instances.

Two ROS nodes were implemented to listen to ROS topics to get poses of objects in the scene, and states of joints involved in a certain task. The outputs from these two nodes are used to populate the knowledge base. Based on that, the spatial reasoning module can be initiated, and its outputs processed in the knowledge base. The outcome is showed in

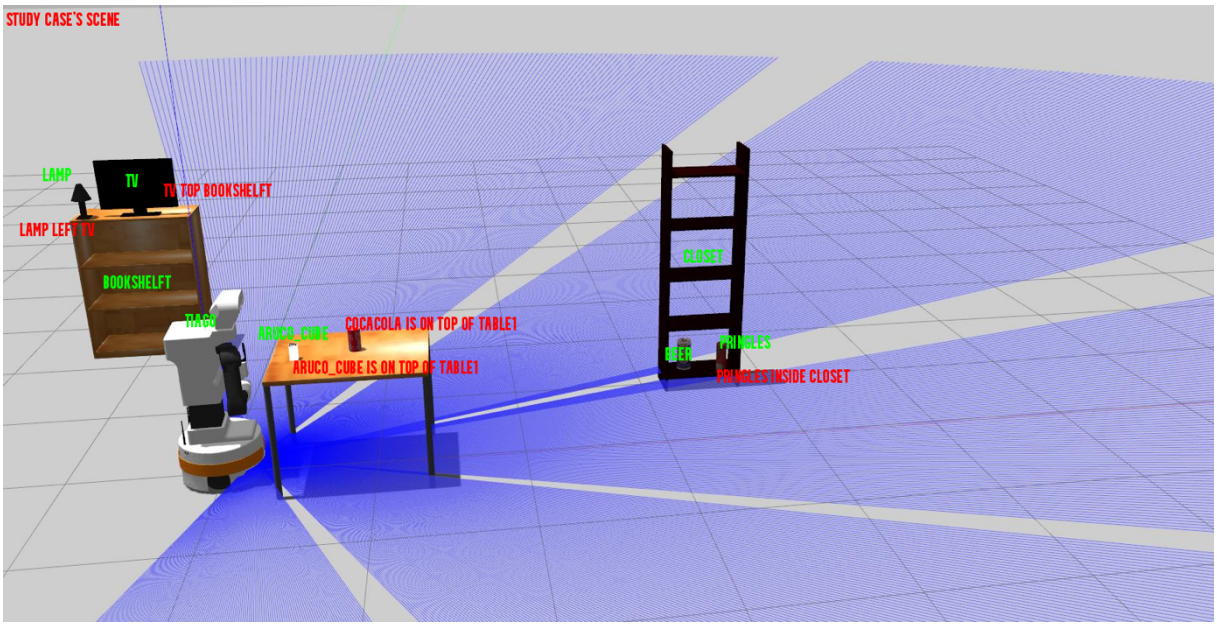



Figure 5 Study Case, spatial reasoning. The texts in green show the object's names while the text in red show object spatial situation w.r.t another object in the scene as saved in the knowledge base.

The body parts of the robot *tiago* (robot's name in the ontology) can be queried easily within the Python bindings by getting all the *part_of* relations in the inferred ontology. This will output the parts of the robot as a python list: *tiago_arm, tiago_gripper, tiago_finger_R tiago_finger_L, tiago_torso, tiago_base, tiago_head,tiago_cam.* Here, we can see that the

left and right fingers are considered as body parts of the robot but in fact they are part of the gripper but since the gripper is also part of the robot, they are automatically classified as part of the robots by the reasoner. This is called auto-classification.

Similarly, objects in the scene can be queried with the simple search query in OwlReady2 library by getting all instances of type Object. The result is a python list of instances of type Object. The data and object properties of each element of the list can be accessed as *element.property* i.e *cocacola.name*, *beer.has_spatial_location*.

The tasks carried out can be queried with a search of instances of type Action, if many, label key in the query can be used to select more specific instances. The goal object, and if possible, the robot that affords (the affordance here is defined as in the previous section) the task can be queried out with direct calls to respective object property. The participating joints can be queried as well calling the object property *has_joint* on the instances, the states of the joints during the task can also be queried using the object property *has_movement*. As this will probably return a long list of states of the participating joints. For specific details such as position, velocity, base link, child link, effort of the joint state, it is possible to call the appropriate data property on top of the joint state instance.

# 5. SUMMARY

An ontology-based knowledge representation package based on ROS stack has been proposed at the end of the project. It can be used to extract deep semantics knowledge on the performed task, the agent and the scene. The work was inspired by some KnowRob's features and built using Python3 binding in order to allow the coexistence of robot control code and knowledge representation.

The work was useful as it helped dive into knowledge representation field, which wasn't familiar to me at the beginning, however with good guidance and research of existing solutions, the path to performing the task become clearer. I learnt to use ontologies specially OWL ontology language as a tool for semantic modelling with Protégé or a Python3 binding, to model manipulation tasks in order to allow the extraction of semantic knowledge, query ontologies using Python3 code. In parallel, it was an opportunity to refresh up my ROS programming skills, and more importantly the usage of docker containers outside of classrooms.

As future improvements, the following can be added to the package: a deeper representation of manipulation tasks following SOMA standard, the implementation of learning methods to allow the use of the knowledge base to predict future actions or states or behaviors for robotic agents, implementation of a parsing mechanism to automatically parse robots' and objects' models from their description files as currently this is done both manually and by subscribing to corresponding ROS topics.

**LIST OF REFERENCES**


[1] "RobotKoop | Forschungsprojekt," *RWU Forschung, Wissens- und Technologietransfer*. https://forschung.rwu.de/en/projects/robotkoop (accessed Feb. 22, 2021).

[2] J. Pages, L. Marchionni, and F. Ferro, "TIAGo: the modular robot that adapts to different research needs," p. 4.

[3] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *Int. J. Robot. Res.*, vol. 32, no. 5, pp. 566–590, Apr. 2013, doi: 10.1177/0278364913481635.

[4] "Ontology:DOLCE+DnS Ultralite - Odp." http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite (accessed Feb. 15, 2021).

[5] J. Bateman, M. Beetz, D. Beßler, A. K. Bozcuoğlu, and M. Pomarlan, "Heterogeneous Ontologies and Hybrid Reasoning for Service Robotics: The EASE Framework," in *ROBOT 2017: Third Iberian Robotics Conference*, vol. 693, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Cham: Springer International Publishing, 2018, pp. 417–428.

[6] M. Beetz, D. Bessler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, and G. Bartels, "Know Rob 2.0 — A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, May 2018, pp. 512–519, doi: 10.1109/ICRA.2018.8460964.

[7] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5, pp. 907–928, Nov. 1995, doi: 10.1006/ijhc.1995.1081.

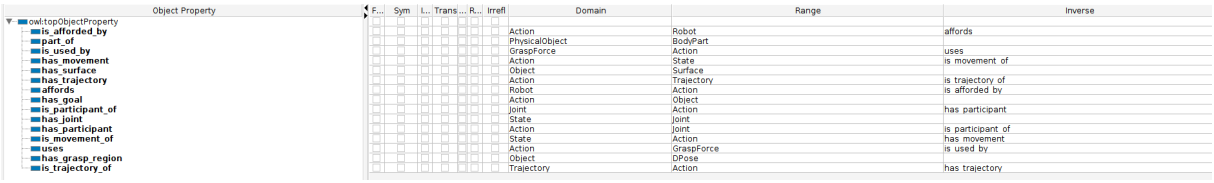
[8] J.-B. Lamy, "Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies," *Artif. Intell. Med.*, vol. 80, pp. 11–28, Jul. 2017, doi: 10.1016/j.artmed.2017.07.002.

[9] M. Diab, A. Akbari, M. Ud Din, and J. Rosell, "PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation," *Sensors*, vol. 19, no. 5, Mar. 2019, doi: 10.3390/s19051166.

[10] T. Tudorache, C. Nyulas, N. F. Noy, and M. A. Musen, "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the Web," *Semantic Web*, vol. 4, no. 1, pp. 89–99, 2013, doi: 10.3233/SW-2012-0057.

## Appendix 1 Class Diagram of Asserted Ontology

# Appendix 2 Property Matrix in Asserted Ontology



| Object Property | F... | Sym | I... | Trans... | R... | Irrefl | Domain | Range | Inverse |
|---|---|---|---|---|---|---|---|---|---|
| owl:topObjectProperty | | | | | | | | | |
| is_afforded_by | | | | | | | Action | Robot | affords |
| part_of | | | | | | | PhysicalObject | BodyPart | |
| is_used_by | | | | | | | GraspForce | Action | uses |
| has_movement | | | | | | | Action | State | is_movement_of |
| has_surface | | | | | | | Object | Surface | |
| has_trajectory | | | | | | | Action | Trajectory | is_trajectory_of |
| affords | | | | | | | Robot | Action | is_afforded_by |
| has_goal | | | | | | | Action | Object | |
| is_participant_of | | | | | | | Joint | Action | has_participant |
| has_joint | | | | | | | State | Joint | |
| has_participant | | | | | | | Action | Joint | is_participant_of |
| is_movement_of | | | | | | | State | Action | has_movement |
| uses | | | | | | | Action | GraspForce | is_used_by |
| has_grasp_region | | | | | | | Object | DPose | |
| is_trajectory_of | | | | | | | Trajectory | Action | has_trajectory |