Semantic Description and Communication of Going around in a Simulated World

DISSERTATION
zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von
M.Sc. Nazeer T Mohammed Saeed

1. Gutachter	Prof. Dr.-Ing. Klaus-Dieter Kuhnert
2. Gutachter	Prof. Dr.-Ing. habil. Madjid Fathi

eingereicht bei der Naturwissenschaftlichen-Technischen Fakultät
der Universität Siegen
Siegen 2019

Tag der mündlichen Prüfung: 07.05.2019

gedruckt auf alterungsbeständigem holz- und säurefreiem Papier

I would like to dedicate this dissertation to my parent.

— **Nazeer Saeed**
(Siegen, 2019)

Eidesstattliche Erklärung

Die vorliegende Dissertation wurde von mir selbständig angefertigt. Die verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt. Eingetragene Warenzeichen und Copyrights werden anerkannt, auch wenn sie nicht explizit gekennzeichnet sind.

Siegen, 2019

M.Sc. Nazeer T Mohammed Saeed

Abstract

In recent years, mobile robots have helped to link predictions, imaginations, and expectations to human life. The tsunami of research on mobile robots mirrors their importance in various fields, such as production, agriculture and medicine. Alongside the innovations in sensory solutions, we are witnessing more intelligent mobile robots performing more challenging tasks by using the massive amount of data gathered from various sensors. The excess of the sensory data increases the demand for processing sensory inputs in an understandable form for both - humans and robots. One approach for developing understandable processing and exchange is the use of semantic technology. Semantic technology is a major technology for building semantic knowledge bases in machine-readable form, with ontologies as a mature, flexible and well researched implementation.

A considerable amount of research shows the existing impact of ontologies in the field of robotics. However, there is still a lack of work in regard to the real-time semantic knowledge acquisition from sensory outputs of different sensors for mobile robots and the use of sensory data for real-time ontology population and consequently natural language communication between humans and robots. This work employs semantic technology in the field of robotics to acquire a better understanding of an environment, navigated and sensed by a mobile robot and to continuously produce semantic information to facilitate communication between other robots and human beings.

In this research, the Resource Description Framework (RDF), which is a semantic web standard, is utilized for the instant creation of semantic information from sensory outputs during navigation with a mobile robot. A novel approach for modeling complex RDF relations has been introduced; it uses a combination of sensory data from various sensors of a mobile robot to model single complex RDF-statements that represent inter-object relations between detected landmarks while exploring the environment with a mobile robot in a way that humans would express it. These statements are then collected and stored in an ontology, hence, a novel, efficient ontology is then designed for the real-time, online population; this is then tested in real-time. The proposed concept utilizes a natural language communication interface to facilitate real-time human-robot communication regarding the navigation and environment that has been explored.

To evaluate the system, a mobile robot has been simulated and equipped with different sensors and placed in a simulated environment to navigate and explore the environment. While exploring, sensory data is collected and processed to model semantic information representing its tour and vision of its environment. The ontology is then populated with this information in real-time and is used by the system to facilitate natural language communication with the robot regarding its tour and the explored environment.

The results show the real-time population of the ontology with RDF-statements created from sensory outputs representing the tour of the mobile robots and the environment in a semantic representation. The efficiency of the system in transforming sensory data into semantic information, the ability of the mobile robot to describe the real-world environment semantically, and also its ability to answer natural language questions regarding its tour and the environment are proof of the soundness of the proposed system.

Zusammenfassung

In den letzten Jahren hat sich die Verbindung mobiler Roboter mit diversen menschlichen Aktivitäten deutlich intensiviert und sie beginnen die in sie gesetzten Erwartungen und Vorstellungen zunehmend zu erfüllen. Die tsunamiartig wachsende Zahl an Forschungsarbeiten im Bereich der mobiler Robotik spiegelt ihre Bedeutung in verschiedenen Bereichen wider, z. B. in der Industrie, der Landwirtschaft und der Medizin. Zusammen mit neuen Entwicklungen in der Sensorik erleben wir, wie immer intelligentere mobile Roboter anspruchsvolle Aufgaben lösen, indem sie große Mengen an Daten verarbeiten, die von verschiedenen Sensoren erfasst werden. Die Flut der Sensordaten erhöht die Notwendigkeit der Verarbeitung sensorischer Eingaben in einer für Mensch und Roboter verständlichen Form. Ein Ansatz für eine verständliche Weiterverarbeitung der Sensordaten ist die Verwendung von semantischer Technologie. Die semantische Technologie spielt eine zentrale Rolle beim Aufbau semantischer Wissensbasen in maschinengeeigneter Form. Für diesen Zweck sind Ontologien eine ausgereifte, flexible und wohluntersuchte Form der Realisierung.

Eine beträchtliche Anzahl an Untersuchungen zeigt den Einfluss von Ontologien auf dem Gebiet der Robotik. Es fehlt jedoch immer noch an Arbeiten, die sich mit der semantischen Echtzeit-Wissenserfassung gespeist aus den Sensoren mobiler Roboter befassen. Ebensowenig ist es bisher üblich Ontologien in Echtzeit aus Sensordaten zu erzeugen bzw. sie mit solchen zu befüllen und damit letztendlich eine Kommunikation zwischen Mensch und Roboter in natürlicher Sprache zu ermöglichen.

Diese Arbeit wendet semantische Methoden auf die mobile Robotik an. Dies dient dem besseren Verständnis der Umgebung, die durch Sensoren und Navigation im Roboter abgebildet wird, und der mitlaufenden Erzeugung semantischer Informationen aus diesen Daten, um mit anderen Robotern und Menschen zukommunizieren.

In dieser Studie wird das Resource Description Framework (RDF), ein semantischer Web-Standard, für die sofortige Erzeugung semantischer Informationen aus den sensorischen Daten während der Navigation mit einem mobilen Roboter verwendet. Ein neuer Ansatz für die Modellierung komplexer RDF-Beziehungen wird vorgestellt. Verschiedene Sensordaten des mobilen Roboters werden kombiniert, um damit komplexe RDF-Aussagen zu erzeugen. Diese repräsentieren Inter-Objekt-Relationen zwischen

Orientierungspunkten in einer Art und Weise, wie Sie ein Mensch aufstellen würde, während sich der Roboter in der Umgebung bewegt. Diese Aussagen werden gesammelt und in einer Ontologie gespeichert. Infolgedessen wird eine neuartige, effiziente Ontologie geschaffen, die in Echtzeit besetzt und getestet wird. Das hier entwickelte Konzept beinhaltet eine natürlich sprachliche Schnittstelle, die es Menschen erlaubt in gewohnter Weise mit Robotern über räumliche Vorgänge zu "sprechen".

Um das System zu evaluieren, wurde ein mobiler Roboter simuliert und mit verschiedenen Sensoren ausgestattet. Dieser Roboter wurde in einer simulierten Umgebung platziert, um in dieser zu navigieren und diese zu erforschen. Während der Navigation werden Sensordaten gesammelt und verarbeitet, um semantische Informationen zu erzeugen, welche die Strecke und die Wahrnehmung der Umgebung repräsentieren. Die vordefinierte Ontologie wird dann mit diesen Informationen in Echtzeit gefüllt. Sie wird vom System verwendet, um die Kommunikation mit dem Roboter bezüglich seiner Umgebung zu erleichtern.

Die in Echtzeit aufgebaute Ontologie -zumeist bestehend aus in RDF-Aussagen gegossene Sensordaten- beschreibt die gefahrene Strecke des mobilen Roboters und die zugehörige Umgebung. Die Effizienz des Systems, Sensordaten in semantische Informationen zu transformieren, die Fähigkeit des mobilen Roboters, seine reale Umgebung semantisch zu beschreiben und die Fähigkeit, Fragen über die Strecke und die Umgebung in natürlicher Sprache zu beantworten, sind Beweis für die Funktionalität des hier vorgestellten Systems.

Acknowledgement

At this moment of accomplishment, first of all, I would like to express my special gratitude and appreciation to my supervisor Prof. Dr.–Ing. Klaus-Dieter Kuhnert, for providing me an opportunity at the Institute of Real-Time Learning Systems (EZLS), University of Siegen. I would like to thank him for providing me with the guidance, support, and freedom to conduct my research; The EZLS Institute provided me a conducive environment to learn and the resources to utilize that resulted in this research; without this, this work would not be possible.

It was my privilege to work also under the supervision of Prof. Dr.–Ing Majdid Fathi, the head of the Institute of Knowledge-based Systems and Knowledge Management (KBS & KM) at the University of Siegen. I would like to thank him for giving me the opportunity to learn from him. He was a supervisor and a mentor to me, and I am grateful to him for reviewing my work, this dissertation, and providing constructive feedback. I would also like to give special thanks to Christian Weber for his assistance, time and careful attention to details.

This dissertation is the end of my journey in obtaining my Ph.D. degree. I have not traveled in a vacuum in this journey. It has been kept on track and been seen through to completion with the support and encouragement of numerous people including my well-wishers, my friends, and colleagues. I would like to thank all those people who made this journey an unforgettable experience for me.

Contents

1 Introduction

Every once in a while, a new technology, an old problem, and a big idea turn into an innovation.

— **Dean Kamen**
(engineer, inventor, and businessman)

1.1 Motivation

The rapid advancement of robotics technology began in the 1950s, and nowadays its presence is undeniable in different high-tech areas such as medicine, manufacturing and production, aerospace, military equipment, and transportation. Robots are progressively replacing human workers and performing different hazardous tasks that humans are unable to do efficiently. Particularly, in the area of production, the presence of robots reduces worker injuries, the number of accidents and waste. In the health sector, there are certain surgeries that, in some cases, have been performed under conditions where the surgeon was far away from the patient utilizing robots through long distance connections. Robotic technology allows surgeons to make subtle and accurate cutbacks and perform surgeries that are impossible using human capabilities alone. In the automotive industry, robotic technology helps to produce cars with more accurate welding and much fewer blemishes. This new level of quality cannot be expected from utilizing only human resources.

Over the past century, robots have been mostly fixed and set in places such as factories, to accomplish isolated tasks flawlessly. In the 21st century, however, attention is shifting towards mobile robots that can use a mobile platform to carry out a wide range of activities in environments that are not accessible by human beings. The mobile robot is referred to as an unmanned mobile electromechanical system that relies on its maneuverability and motion to perform a specific mission or operation by receiving information from its sensors.

Robots have indeed become quite intelligent and are now physically capable enough to find their way out of factories to drive and walk among us. With the innovations and advances in artificial intelligence (AI), they have gained even greater social abilities.

These innovations have led to the development of new mobile robots, and fusing of a wide range of sensory inputs for acting in their environments. A vast number of these sophisticated robots have currently populated the world; we can find them in almost every section of our modern society. However, with these achievements, new challenges have been raised in the robotics field. One of the most important challenges is the robot's ability to understand its environment. It is not possible to pre-program a mobile robot for each and every situation they might encounter in their new world. They have to be able to gain understanding on their own using models to abstract the outer world and use the derived understanding for decision making and finally communication between robots and between robots and humans. The complexity of the development of means of communication between robots as well as between robots and human, spawned some new fields, such as Robot-Robot Interaction (RRI) and Human-Robot Interaction (HRI). Yet, models to support hitch-free communication are still scarce.

Due to the diversity of multi-robot applications in different areas, such as exploring remote environments and production lines in factories, the importance of RRI increases. In multi-robot applications which are also referred to as Multi Agent Systems (MAS) multiple robots are required to work together to perform a specific task. For the success of such tasks it is crucial for them to be able to interact with each other. Moreover, in most if not all of the cases robots are controlled by a human operator, and need to actively interact with humans to perform their tasks. The interaction between robots and humans becomes more critical when robots are operating in a shared environment with humans (Losey et al., 2018), potentially even cooperatively.

This rapid progress of research and development indicates a promising future in which humans and robots will be sharing the same environment (Prestes et al., 2014). However, as robots continue to enter our world, the challenge of communication with them raises accordingly. The challenge is that human beings do not understand robot languages and vice versa. It is not only complicated to transfer information between the human and the machine, but machines are also facing many problems in transmitting information among themselves. Consequently, there is a great need for a formal language which robots can understand and use to communicate with one another as well as, a language which robots can use to express their understanding of a situation and finally use this to communicate with humans. Hence the Partnership for Robotics in Europe highlights communication as one of the fundamental elements of intelligent robots (Robotics SPARC, 2016). Robot communication with humans eases their entrance into the lives of humans, while, the communication between robots raises their capability to perform in multi-robot applications. This increases the need for formal knowledge representation which involves developing a standard that is not yet represented in the existing literature.

The Semantic Web (SW) can address this problem, the idea is to model information in a semantic representation that machines and humans can work on together. Structured data is what machines are able to process, in contrast to complex information and information contexts that are processed by human beings, through a process of understanding. For a system in which humans cooperate and work alongside machines, it is necessary for humans to present information in a way that machines can understand and process, while the machines must also be able to present data in a way that humans can understand. The purpose of SW development is to achieve such a system by using the technologies and standards developed and implemented by the World Wide Web Consortium (W3C). In other words, in the SW, machines have the ability to understand data and participate in the process of analyzing and extracting the information desired by humans in an intelligent manner.

To this end, ontologies, as a basic building block of the SW play an essential role in enabling semantic and reliable data integration as well as information exchange between machines which can be used to express information in an understandable human way (Schlenoff et al., 2012; Carbonera et al., 2013). However, the application of semantic technology in robotics for transforming sensory outputs while exploring environments and also real-time semantic environment description and online ontology population as well as the use of dynamic ontologies populated in real-time by a mobile robot for natural language communication are challenging tasks which are not yet represented in the existing literature.

This dissertation addresses these challenges and describes the application of semantic technology in robotics. This work explores how the sensory data of a mobile robot can be employed for the creation of semantic information that describes the real-world environment, how to populate ontologies in real time while exploring the environment and using the ontology to understand the explored environment and finally how to communicate with the robot in natural language.

1.2 Research questions

This dissertation addresses the following research question:

- **Research question 1:** How can sensory inputs of a mobile robot be transformed into information and be modeled in a human and machine interpretable way to describe the real-world environment semantically in real-time while exploring the environment?

- **Research question 2:** How can semantic information representing the tour of a mobile robot and its perception of its environment be created and collected in real-time using semantic technologies?

- **Research question 3:** How can the collected semantic information be used for a natural language communication with the robot in a system implementation?

To answer the mentioned research questions, the application of semantic technology in the field of robotics has been investigated. Consequently, the following main topic areas have first been studied in detail and their actual state is described in the state-of-the-art.

- Robotics

- Artificial intelligence and simulation

- The semantic technology

- Natural language communication

As shown in Fig.1.1, each of the above-mentioned topic areas is subdivided into their subtopics. For instance, in the robotics field, the application of mobile robots for navigation and localization has been studied. The use of a semantic sensor for semantic segmentation and object detection has been studied, and the process of information acquisition from sensory data has been investigated. To employ the semantic sensor, the application of simulation in AI and particularly in robotics have been studied and utilized. In the area of data modeling and semantic knowledge representation, the use of semantic web technology, and precisely RDF data modeling and ontologies for knowledge representation, have been studied and their applications in robotics have been implemented. In the area of natural language communication, different QA systems have been introduced, this work, however, implemented their application for communication with the mobile robot.

As a result, this work proposes a novel concept that answers the mentioned research questions. The concept is the result of a broad investigation of the use of ontologies for semantic knowledge representation and communication of information in the field of robotic. The details about the proposed concept is provided in section 4.1.

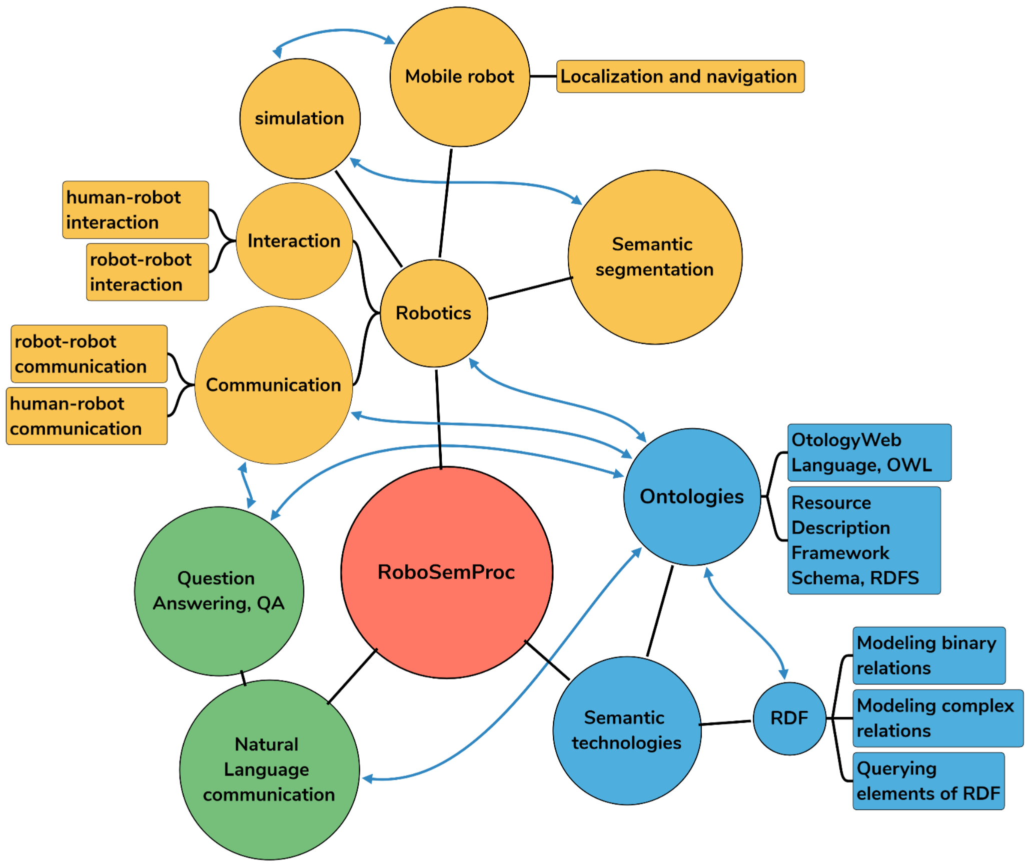


Fig. 1.1: The investigated connection of topics to the proposed concept

1.3 Dissertation outline

This section provides an overview of this dissertation, and the structure of the dissertation is illustrated in fig.1.2 and explained in the following. The dissertation is comprised of six chapters, including this introduction (Chapter 1) which provides the motivation, research questions and an overview. The remainder of the dissertation is as follows.

Chapter 2: The state of the art

The system proposed by this work is the result of a combination of different system software and concepts. It utilizes the semantic technology in robotics to answer research questions. This chapter describes the state-of-the-art of the connection fields and presents related works.

Chapter 3: The technical settings of the proposed system

This chapter describes the technical settings of the proposed system. It introduces the Autonomous Mobile Outdoor Robot (AMOR) and describes the process of simulation.

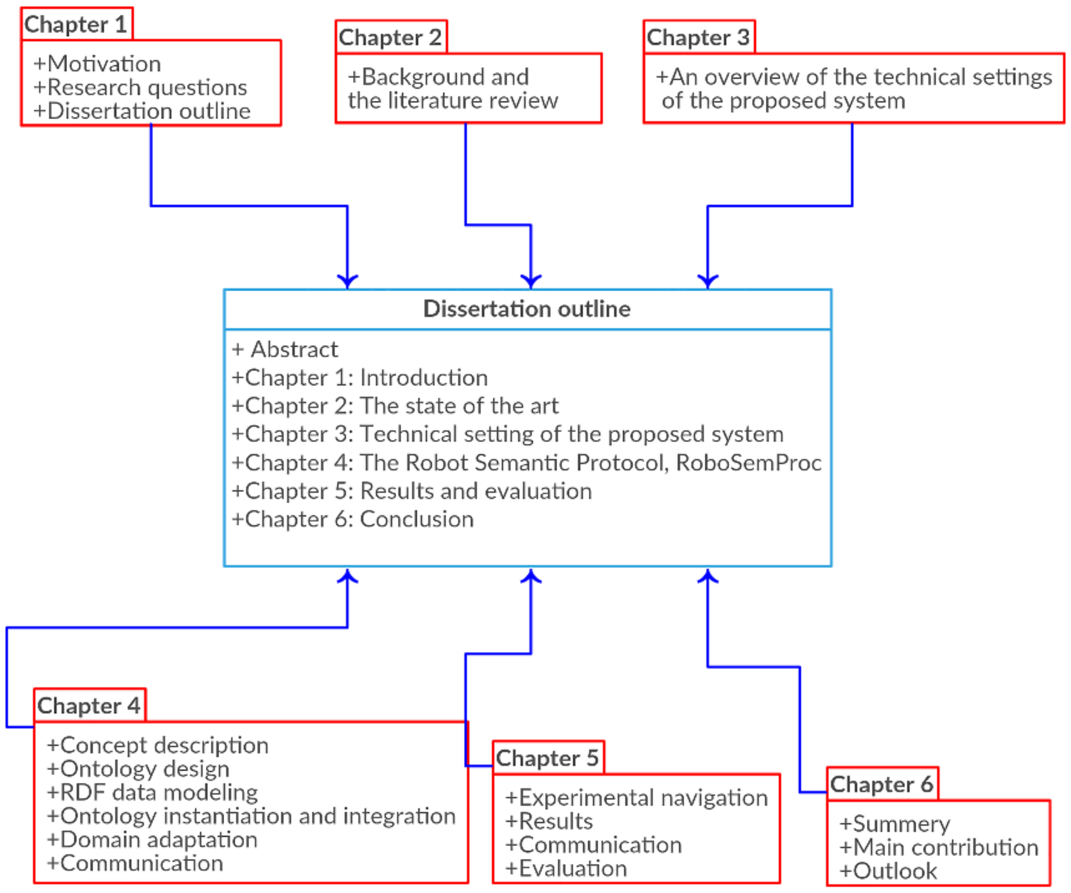


Fig. 1.2: An overview of the structure of the dissertation

Moreover, the chapter provides detailed information about different sensors of AMOR, and the process of object annotation and sensor data processing.

Chapter 4: The Robot Semantic Protocol, RoboSemProc

This chapter describes different phases of the proposed system called RoboSemProc, starting from ontology design, into Resource Description Framework (RDF) modeling, ontology instantiation and integration, domain adaptation and natural language communication. Moreover, this chapter presents different approaches for modeling RDF-statements. This work proposes a new approach for modeling RDF-statements. A comprehensive description is provided in this chapter which illustrates the performance of each approach.

Chapter 5: Results and evaluation

As its name indicates, this chapter discusses the results and evaluation of the RoboSemProc. For this reason, an experimental navigation has been conducted, in which AMOR explores different areas of its environment and uses its sensory data for the creation of semantic information that represents its tour and its vision of the environment. This chapter shows the result of real-time ontology instantiation and the use of populated ontology for natural language communication. Moreover, it presents the evaluation of the different stages of the proposed system.

Chapter 6: Conclusion

This chapter is the last chapter of the dissertation in which a summary, the main contributions and the outlook of the research are provided.

2 The state-of-the-art

I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.

— **Alan Turing**
(mathematician, logician, cryptanalyst and computer scientist)

2.1 Robotics

To find the origin of the science of robotics, we have to make a journey in time, back to the earliest times of ancient Greece, when the first animated sculptures were made, in the first century BC. One of the long-term desires of humankind was the use of inhuman servants who could work 24 hours a day without tiredness and constant control, as well as have no human problems or limitations. Since such an act was like a dream in the past, at that time in theaters and plays, these heavenly slaves were called robots. About 100 years ago, Czech Slovak writer, Carl Čapek, used the word Robota, which means "worker" or "tedious effort", in the play Rossumovi Univerzální Roboti (R.U.R). In this play, the power of his mental machine was far more than human. As a result, at the end of the story, it overcame its creators. The word robotic was also invented by a writer. The Russian-born American scientist, Isaac Asimov, first used the word in 1942 in his short stories. Asimov's view of robotics was much sharper than Čapek. Asimov also presented his famous proposal, *the Three Robotic Laws* (Clarke, 1993).

Robotic law posed by Asimov:

1. Robots should never hurt humans.

2. Robots shall execute the orders of humans without conflicting with the first law.

3. Robots must protect themselves without violating the first and second laws.

These fictional ideas remained a dream until, in 1941, Konrad Zuse, the inventor of the world's first programmable computer, introduced his innovative invention, the Z3 which was the first electronic computer, later in the same year he introduced the Z4 which was the first commercial computer. Later in 1946, John Mauchly's introduced the electronic computer, Electronic Numerical Integrator and Computer (ENIAC) at the University of Pennsylvania (Eckert and Mauchly, 1964). Then, the first universal digital computer ENIAC solved its first issue at Massachusetts Institute of Technology (MIT). Finally, in 1959, Marvin Minsky and John McCarthy founded the first artificial intelligence lab at MIT.

From the definitions of the past, we note that today the word robot is used for any human-made machine that can do what is normally done by humans. All devices used in automotive, electronics or underwater exploration are considered a kind of robot. Currently, robots are used in various branches of industry and these robots are able to automate the production of various tasks at various locations of the production lines in an anticipated manner. In the future, these advanced machines will probably be used to care for children and the elderly, for example, they will be language teachers, pre-services, courier services and deliver goods or maintaining security, and perhaps even the task of controlling health care and medical tests.

2.2 Mobile robots

A mobile robot is referred to as an unmanned mobile electromechanical system, which relies on its maneuverability and motion and by receiving information from its sensors it performs a specific mission or operation. For mobile robots, sensors are tools for connecting them to the outside world and obtaining environmental and internal information. The robot shall be able to use its sensory data for understanding, decision making, path planning and communication between other robots as well as human beings. Autonomous cars are excellent examples of mobile robots. Most automotive companies attempt to produce fully automated or semi-automatic cars. Depending on their motor system and maneuverability mobile robots are categorized into the following categories:

1. Air-based robots such as Unmanned Aerial Vehicles (UAVs) and dual rotor helicopters.

2. Land robots such as wheeled robots and tracked robots.

3. Water-based robots such as swimming robots, submarines, and submersibles.

2.2.1 Localization and navigation

The navigation of a mobile robot is the science of determining the position of a robot and plotting a path for safe exploration and movement from one point to another. According to Barrera (2011) the mobile robot navigation includes the following outlined activities:

- Perception: Interpreting and processing the obtained sensory data.
- Exploration: Strategies guiding the mobile robot to select the next path to take.
- Mapping: The development of a spatial illustration by interpreting the sensory data perceived.
- Localization: The process of estimating the robot's position in the environment.
- Path planning: The strategy to find an optimal path toward the destination.
- Path execution: The adaptation of the motor actions to environmental changes.

In „Localization and Navigation," 2008 the importance of localization and navigation is stated as follows,

> "Localization and navigation are the two most important tasks for mobile robots. Of course these two problems are not isolated from each other, but rather closely linked. If a robot does not know its exact position at the start of a planned trajectory, it will encounter problems in reaching the destination."

The importance of navigation and localization increases accordingly with the increase entrance of robots into different areas of human life (Bräunl, 2008, p. 241).

Navigation and obstacle avoidance are two primary tasks of a mobile robot. Due to the increasing implementation of mobile robots in human life and the physical space that they share with humans, it is vitally important for a mobile robot to produce a navigation behavior that is not only safe but also human-like. According to the work of (Kruse et al., 2012) "human-like navigation in general refers to a robot's motion patterns that appear natural and intuitive to human or the robot's ability to behave in such a way that a human partner does not feel aggravated or afraid by the robot's movements and can easily infer the intentions of the robot." Although most of the existing approaches in literature deal with safe robot navigation, it has been proven that mobile robots with human-like behavior are more likely to be considered socially acceptable by human beings, and a number of researchers considered enabling a robot to perform safe navigation with human-like behavior at the same time.

In general, navigation can be categorized into global planning and local planning. In global planning algorithms, a mobile robot uses existing environment information for path planning (Ferguson and Stentz, 2006). As for local planning algorithms, the path planning in the dynamic environment works based on sense-act process (Minguez and Montano, 2004). Both of these global and local planning algorithms are ensuring the safety for path planning and navigation, however, they are not natural and socially acceptable. Many studies address this gap, and some new approaches have been introduced for a safe and human-like navigation behavior.

One approach is learning human-like navigation from observations of pedestrian trajectories, Kuderer et al. (2012) presents an approach of predicting movements of pedestrians. He applied a maximum entropy feature based learning method to capture all relevant aspects of the pedestrian trajectories for determining the possibility of distribution that underlies human navigation behavior. The approach has been applied to a mobile robot for predicting future interactions with pedestrians in a socially compliant way (Kuderer et al., 2012). In another approach, presented by Guzzi et al. (2013) the same heuristics for common avoidance used by human beings are employed to create a fully-distributed algorithm for mobile robot navigation. The navigation algorithm used by this approach which was based on an obstacle avoidance heuristic has modeled pedestrian behavior well. The resulting trajectories have been human-friendly since they could be predicted by human beings, providing a suitable algorithm for a mobile robot that shares navigation spaces with human beings. The work of (Pradeep et al., 2016) presents another navigation algorithm for a mobile robot for a human-like navigation behavior. This work enables the mobile robot to perform safe navigation in an unknown dynamic environment based on its sensor inputs. Robot Proxemics-based Navigation (RPN) is introduced to the robots for decision making based on environment information. Pedestrian studies have enriched this proposed algorithm in order for the mobile robot to be able to navigate in unknown dynamic environments. The algorithm works based on a cognitive representation of the environment, and it does not need to track any obstacles or make assumptions on obstacle shape and velocity. The outcome of this study resulted in human-like navigation behavior (Pradeep et al., 2016).

2.2.2 Human-robot interaction

Due to the diversity of multi-robots applications in different areas, such as exploring remote environments and production lines in factories, the importance of RRIs increases accordingly. In multi-robot applications MAS, a number of robots are required to work together to perform a specific task, and, the success of such tasks is highly depending on the interaction between them. Moreover, in most if not all cases, robots are controlled by an operator, and they need to interact with humans to perform their tasks. The interaction between robots and humans becomes more important when robots are operating in a shared environment with humans (Losey et al., 2018).

Goodrich and Schultz (2007) best define HRI, they refer to HRI as "a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans." (Goodrich and Schultz, 2007, p. 204), they also define interaction as "the process of working together to accomplish a goal. Interaction, by definition, requires communication between robots and humans." (Goodrich and Schultz, 2007, p. 217). Additionally, they introduce five attributes that influence the interactions between humans and robots, these attributes are:

- *Level and behavior of autonomy.*
- *Nature of information exchange.*
- *Structure of the team.*
- *Adaptation, learning, and training of people and the robot.*
- *Shape of the task.*

Furthermore, they distinguish between different types of human-robot communication by classifying them into two main categories, remote interaction, and proximate interaction. In remote interaction, the human and robot are temporally or spatially separated, for example, a robot exploring in a remote environment. As for proximate interaction the human and robot are co-located by sharing the same environment, for example, service robots (Goodrich and Schultz, 2007).

One example of HRI is presented in the work of Liu and Nejat (2013) where an overview of a single robot as well as multi-robot for urban search and rescue (USAR) environments have been presented. They have proposed different approaches for semi-autonomous, multi-robot and low-level robot control, to enhance the performance of rescue robot in the cluttered USAR and to minimize the workload of robots operators, however, the importance of the HRI is undeniable (Liu and Nejat, 2013). Whereas Saez-Pons et al. (2010) address the importance of HRI in a multi-robot team by assigning a robot team to perform a task with non-expert firefighters in a real-life situation. In their scenario robots needed to work with humans and be able to interact with them.

Research concerning HRI is not limited to the land. In fact there have been applications underwater, as in the work of Kulkarni and Pompili (2010) where a group of autonomous under water vehicles perform an allocation task underwater by utilizing an allocation framework proposed for autonomous underwater vehicles. The common assumption in most of the multi-robot teams is that they are either autonomous or controlled by an operator. A hidden assumption is that, in most of the cases, robots cannot perform smoothly and overcome failures without a human being involved in one way or another, after all robots are used by humans (Rosenfeld et al., 2017).

Considering that human interaction is essential for robots to perform their tasks underwater, exploring in remote environments or even working in firefighting scenarios when few trained people are involved, one can conclude how important it can be for a robot to operate within a shared environment in our houses, workplaces or even at schools or hospitals.

According to Dautenhahn (2007) interaction in the HRI research is classified into three approaches namely: robot cognition-centered view, human-centered view, and robot-centered view see Fig.2.1, (Barrera et al., 2017; Dautenhahn, 2007), The HRI approaches are described as follows:

- Robot cognition-centered view: the robot is an intelligent system and able to solve major problems.

- Human-centered views: the human's perspective is emphasized and taken into consideration for the design and behavior of the robot.

- Robot-centered views: the robot, as an autonomous entity, is in the center and the human, as a caregiver, needs to respond to the robot's need.

2.3 Semantic segmentation

Nowadays, in our modern society with the presence of a diversity of intelligent devices and machines, the problem of communication becomes more evident. How can intelligent machines and devices communicate among each other and how can they respond

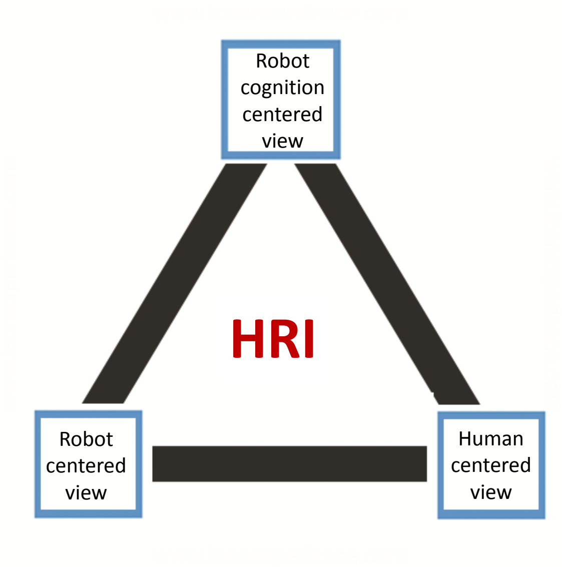


Fig. 2.1: The three conceptual approaches of social interaction in the HRI, modified from (Dautenhahn, 2007) and (Barrera et al., 2017).

to human's questions and communicate with them? Would it be possible for two intelligent individuals to have meaningful communication without a common language and understanding? How can human and machine communicate information?

To answer these questions, and to communicate information, several systems and search engines have been developed to search and provide answers for the desired information and also to present ways and means of communication between human and machines. Regardless of their advantage, in most cases, these technologies are rendering low-level communication. One good example is the Google search engine, in which the system is presenting information by merely comparing the syntax of the questions asked without considering its semantic.

In semantic communication, the machine is provided with the ability to understand the semantics of the information this enables agents to communicate more efficiently which provides a higher level of fast and accurate means of communication.

"The building is behind the tree" is an example of how humans describe a scene. Breaking down images into distinct entities is the key to understanding images, and it helps us to reason about the behavior of objects. Place classification, object detection and recognition, semantic segmentation and labeling are some pre-requirements of creating a semantic map that uses sensory data. The reasearch of Stückler et al. (2015) and Filliat et al. (2012) are some examples of object recognition, where a map of objects or semantic Simultaneous Localization and Mapping (SLAM) has been presented by the combination of SLAM with object recognition for Red Green and

Blue (RGB) and a Red Green Blue Depth (RGBD) sensor. They proposed an object detection approach which was based on a color and depth camera that combines 3D, texture information and color within a neural network for robust object detection. Their results show that RGBD object recognition improves the object recognition performance. However, the object segmentation was restricted to isolated objects.

With advancements in image processing and object detection methods, we can, of course, draw bounding boxes around individual entities in the image. However, to gain a real understanding of a scene from a human perspective, we need the labeling of the boundaries for every entity at pixel-level precision; this becomes vitally important in automotive industry and mobile robot technology that requires a clear understanding of the objects in their surrounding (Andy and Chaitanya, 2017).

Since the semantic segmentation is the process of labeling images in pixel level it is more accurate and applicable than other object detection methods. Therefore, semantic segmentation received much attention in the industry, and research in various areas such as:

- Robotic vision: object recognition, semantic map labeling, and navigation.
- Autonomous driving: path planning, detecting obstacles such as vehicles, constructions, pedestrians, sidewalks.
- Medicine: investigating and identifying cancerous anomalies.
- Industrial inspection: detecting defected elements in the production line.
- Satellite imagery and photogrammetry: identifying geographical areas in the images such as rivers, mountains, and deserts.

Recently, convolutional neural network (CNN)s have faced a rapid advancement in computer vision, particularly in image classification, object detection and recognition, and most importantly semantic segmentation (Girshick et al., 2014). The success of CNNs relies on their ability to learn rich feature representations. Recent semantic segmentation algorithms are frequently formed to solve the problem of structured pixel-wise labeling based on CNN (Chen et al., 2014). They convert an actual CNN architecture for classification to a fully convolutional network (FCN). One example of such system is the work of Long et al. (2015) in which the problems of semantic segmentation and classification have been addressed by CNN. In this research, they introduced FCN which takes an input of random size and produces a correspondingly-sized output. Their results show impressive improvements in object recognition and semantic labeling that is needed for the semantic mapping system which can be used by a mobile robot for safe navigation.

Due to their importance, object recognition, localization, and semantic segmentation have gained the attention of the researchers in both academic and industry sections. The importance of these fields gained the attention of some of the most famous world competitions. In the PASCAL visual object classes (VOC) 2012, a number of innovated works in different areas such as object detection, object classification and semantic segmentation with 200 different categories of objects have been presented (Everingham et al., 2012). In the Imagenet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014), the image classification and localization challenge with 1000 categories have been conducted where the competitors presented their studies and works in the areas of the object recognition, detection and localization (Russakovsky et al., 2015). One of the most current and state-of-the-art work for object detection and classification is the work of (Redmon et al., 2016) in which the authors introduce a robust object detection system You Only Look Once (YOLO) which is a unified model for the real-time object detection. The proposed system has been trained on a loss function that directly corresponds to detection performance, and the complete model is trained simultaneously. Fast YOLO is considered the quickest general-purpose real-time object detection system available.

Taking into consideration these developments it seems to be clear that in the near future powerful rather universal semantic segmentation and object recognition systems for natural environments will be available. In this thesis, we focus on high-level interpretation and communication and thus simply assume the existence of such a semantic object recognition module. Looking into this problem in detail would have been a completely different topic and would have blown up the frame of this thesis.

2.4 Simulation in robotics

Simulation technology is a process that helps organizations to predict, compare and optimize the results of a new process or system, without cost and risk of changing the current processes and implementation of the new one. It is one of the most influential methods and available tools for managers, industry engineers, system analysts, etc., which enables them to decide more easily on any manufacturing or service system before it is made. The goal is to create simulations as an analysis tool to predict the impact of existing system changes and design to predict the performance of the new system. There are many definitions of simulation, but Shannon (1998) provided the most comprehensive and complete one. He defines the simulation as "*the process of designing a model of a real system and conducting experiments with this model for the* *purpose of understanding the behavior of the system and/or evaluating various strategies for the operation of the system*".

Simulation may also be the process of real-time modeling and testing the model under certain conditions to evaluate and predict its future behavior. There is a high demand for using simulation in the following areas:

- Training: the high risk of practice on a real model for beginners like airplane guidance and patient surgery.

- High-speed: the execution time of the simulated model is higher compared to normal speed of the actual model.

- Low cost of simulators: the real model may not be available due to high cost.

- Lack of a system for real-world testing: the real sensor has not yet been created.

The robot industry always seeks to maximize the productivity of facilities to achieve predetermined goals. Therefore, simulation can be used as one of the most effective and powerful techniques which makes it possible to create a system and examine it to obtain conclusions about the real-world system's performance. Thus, the advantages of the use of simulation in robotics and artificial intelligence become evident. Kehoe et al. (2015) used simulation in their study on cloud robotics and automation. Duarte et al. (2014) introduced an open-source simulation platform *JBotEvolver*, for research in evolutionary robotics. Another example is the work of Qu and Yuan (2015) where the MATLAB robotics toolbox was used to simulate foaming mold cleaning robot.

Hanna and Stone (2017) used the Grounded Simulation Learning (GSL) framework by introducing a new algorithm Grounded Action Transformation (GAT), for evaluating real-world scenarios in simulation. Furthermore, simulation has been used in a neurorobotic platform of the human brain (Roehrbein et al., 2016). Due to its simplicity and efficiency, simulation technology has been used almost in all areas of robotics. According to the Robotic Industries Association, the advantages of using simulation in robotics are as follows:

- Proof of design: Usually, robots are not one-size-fits-all. The simulation, therefore, attends to ensuring that the design of the robot will fit its purpose.

- Proof of process: The simulation ensures that the robot will be effective, and it will achieve the planned task efficiently in a specific period.

- Maximize your investment: Due to the low cost and higher speed of execution in a simulation, the time and money spent are less compared to its alternatives.

Due to all these advantages, we have chosen to implement our system in a simulation environment. Especially the cost-effectiveness and the ability to build a very elaborate system motivated this approach.

2.5 The semantic web

The semantic web (SW) is, in fact, an extension of the current web that enables the computer and individuals to collaborate better with each other. The idea is to have well-defined and connected data on the web which it can be used to integrate, automate, discover, and reuse data through a variety of applications. In the initial web of inhomogeneous agents, there was no machine-understanding of the information gathered, and web contents were not interpretable for machines. The SW, is on the other hand an evolution of a web that converts human-understandable documents into those that are machine-understandable. As a result, the ability to collaborate has increased the range of web applications. The SW is a new architecture for the World Wide Web (WWW), which brings together traditional web content with an understandable machine form. The main motive for the SW is to increase automation, information processing and improve interactions and collaboration among information systems. Therefore the information provided in the SW should be thoroughly dynamic so that SW capabilities can be fully exploited. Hence, the presentation of the meaning of data and dynamics are two main attributes of the SW (Berners-Lee, 2001).

One of the key issues in the SW discussion is the data approach manner. The basic point is that human beings do not understand machine language and vice versa. Consequently, as previously introduced, it is not only challenging to transfer information between the human and machine, but machines are also faced with many problems in transmitting information among themselves. The SW addresses this problem; the idea is to model information in a form that enables machines and humans to work together. Berners-Lee (2001) best known as the inventor of the WWW, has defined the SW as " an extension of the current web in which information is given well-defined meaning better enabling computers and people to work in cooperation". He further expressed his vision of the SW as follows:

> "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms

> of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize." (Berners-Lee, 2001, p. 1).

Data is what the machine is able to process, versus the information that is processed and understood by human beings. For a system in which humans cooperate and work with machines, it is necessary for humans to present information in a way that machines can understand and process, and simultaneously the robot must present data that humans can understand. The purpose of SW development is to achieve such a system by using the technologies and standards developed and implemented by the W3C. In other words, in the SW, machines (robots, servers, and computers) have the ability to understand data and participate in the process of analyzing and extracting the information desired by the human in an intelligent manner. Consequently, machines will be able to communicate with each other, as well as with humans. To illustrate, when a robot reads a sentence, it must be able to grasp the grammar of the sentence, that is, it needs to understand the subject, predicate, and object in that statement, and when reading a paragraph, it should be able to recognize the relation of the sentences to each other (Web, 2015).

To achieve this goal, a particular architecture has been developed and presented by W3C. Based on a philosophy of progress, a step-by-step framework has been designed from the bottom to the top. Different layers of the SW architecture are shown in Fig.2.2. RDF, is the base language used in the SW, is premised on Extensible Markup Language (XML). XML is also based on Unicode and Uniform Resource Identifier (URI). Therefore, it supports various languages. The main part of the SW is the ontology that describes the SW entities. At the top of the ontology, there are rules that allow the derivation of new knowledge from existing knowledge. By creating a standard framework for existing rules, we can prove the consistency of knowledge and share this proof in different applications. The proof layer of the SW executes the provided rules and together with the trust layer mechanism it evaluates whether to trust the given proof or not (Berners-Lee, 2001).

The SW presents tools and technologies to provide a complete solution for modeling semantic information. The diversity of application and use of the SW in different areas such as semantic web, Internet of Things (IoT) and AI is proof of its strengths in modeling semantic information. The SW with its set of standards and recommendations such as RDF, RDFS, OWL, SPARQL, rule languages, and proof and trust layers is considered the only comprehensive solutions for modeling the complete semantic system. The application and use of its alternatives such as JSON-LD and schema.org are limited to only modeling semantic information for isolated applications.

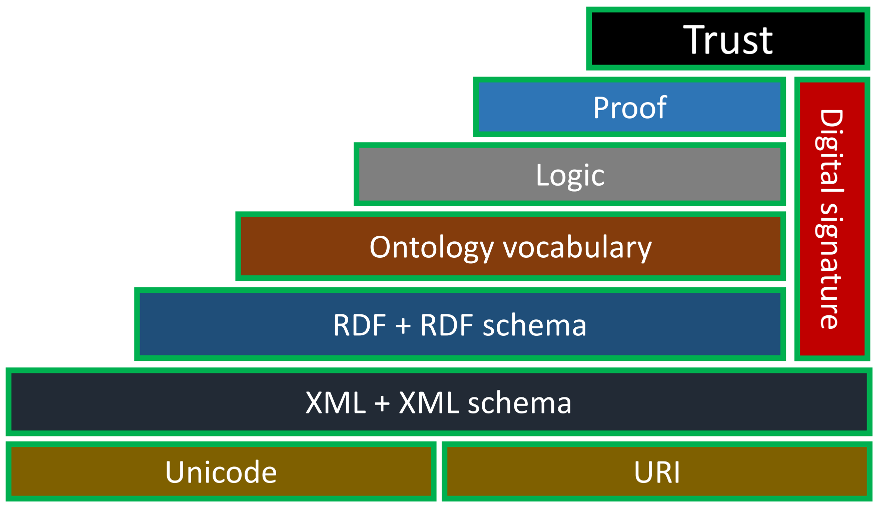


Fig. 2.2: The SW layers, modified from (Web, 2015).

2.5.1 Resource description framework, RDF

Consider reading the following sentence in a web page: "Albert Einstein, the famous theoretical physicist, the author of The World as I See it, was born in Germany in 1879". Look at how profound this sentence is, there is the name of the author, his profession, book, place, and date of birth, but unfortunately, the computer sees this statement as just a series of "words". Although the sentence contains five separate pieces of information, for a computer it has no meaning at all. To get a better idea, consider understanding the following sentence.
有名な理論物理学者のアルバート・アインシュタインは、私がそれを見ると1879年にドイツで生まれました。
Assuming you cannot speak Japanese, this is how a machine sees and understands the sentence; the sentence mentioned above is a Japanese version of the same English sentence in the first example. The level of complexity arises when the machine can not even recognize the words. For instance, in the Japanese sentence, you can not even recognize the words without having some specific knowledge: e.g. if a *hiragana* is followed by a kanji it might be the beginning of a word; whereas if *hiragana* changes to *katakana*, it might be the beginning of a word. Because there are three writing systems (hiragana, katakana, kanji) combined in Japanese. In the SW, machines must fully recognize and understand the meaning behind sentences and not deal with them as just a simple series of words. To achieve this goal the SW utilizes XML and introduces RDF for modeling information.

XML is a markup language like Hyper Text Markup Language (HTML), but unlike HTML, it is designed to describe rather than display information. XML is a simplified subset of the Standard Generalized Markup Language (SGML). SGML is a generic and complex language for data markup, which was created in the 1980s, and because of its simplicity, the XML superseded SGML. XML files are text-based and they can be edited in text editors. A file in XML consists of two parts, a text and markup symbols. The text part contains the main data, while the markup elements represent meta-data of the data presented in the text part of the file.

XML, on the one hand, makes it possible for a human to understand the meaning of the data by sorting them in text format and using sign markers, and on the other hand, the structure and markup used by XML make it easier for a machine to understand and process the data. It is worth mentioning that XML itself does not have specific tags, which makes it possible for the developers to extend the language by defining their own tags. To illustrate the above-mentioned sentence is shown with XML tags in a subsequent XML code section.

Listing 2.1: XML tag example to describe the above-mentioned example.

```xml
<--! Albert Einstein , the famous theoretical physicist , the author of
    The world as I see it , was born in Germany in 1879. -->
<sentence>
   <name> Albert Einstein </name>
   <profession>theoretical physicist </profession>
   <birth>
      <place> Germany </place>
      <year> 1879 </year>
   </birth>
   <book> The world as I see it </book>
</sentence>
```

Using XML, we can tag different entities and their attributes, but we can not express their meaning. For instance, when using XML, if someone marks a price tag with PRICE and someone else marks it with COST, it is not possible to verify that price and cost are synonyms and referring to the same thing. Moreover, the freedom of using tags in XML raises the complexity of querying such data which is very important for a proper human-machine interaction.

As mentioned earlier, the main purpose of the SW is that the data on the web is also understandable for machines. Nowadays, a vast amount of web data is located within HTML, XML documents, such data may be meaningful to humans, but machines can only read and process them without understanding the meaning. Of course, these machine capabilities are greatly appreciated, but the SW is drawn in such a way that the role of machines on the web becomes more evident. One way to add meaning to the data available on the web is to add metadata. Since the machine must understand the meaning of the data, we need languages that can formally state the metadata.

Although the introduction of the SW was first made in 2001, the formation of the SW and the research and development in it is more ancient. In 1994, at one of the first conferences sponsored by the W3C, Berners-Lee et al. first introduced the idea of using semantic technology on the web, but at that time the idea was not well understood (Berners-Lee et al., 1994). Like any other technology, the SW has required the development of standards that could be implemented in a practical and widespread manner. One of the first steps was to provide a standard way to add meaning and describe the data on the web in such a way that the machine could read and understand them. In 1995, efforts were made to develop a technology that could be used to provide more information about data by adding meta-data (data about data), which ultimately resulted in the initial development of RDF in 1998. When in 1999, a more explicit form of the SW was established and its significance became apparent (Marin, 2006).

RDF was the first attempt to add meaning to the data. RDF is an XML-based language that has been developed to describe concepts and create documents in a SW. RDF documents contain descriptions of information in the SW so that they can be understood by machines. In the SW, RDF is used to describe anything physical or abstract that has a unique identity. To describe a resource, we must first define it or so name it. With an identifier, a resource can be uniquely identified and be differentiated from other resources. RDF uses the URI to uniquely identify resources of any kind. URI can be defined as a string of characters used for identifying any name or internet resource. URI consists of two parts: Universal Resource Locator (URL) and Uniform Resource Name (URN), URN specifies the source name whereas URL determines the access method to the resource. The structure and an example of URI are as shown below:

```
scheme:[//authority]path[?query][#fragment]
```

$$\underbrace{\text{https}}_{schema}\text{://}\underbrace{\text{www.eti.uni-siegen.de/}}_{authority}\underbrace{\text{ezls/index.html?lang=de/}}_{path}\ \underbrace{\text{Amor}}_{fragment}$$

RDF was designed to describe resources and the relationship between them. A resource, in general, can represent anything such as an object like a robot, an URL, or even a thought or an idea. The basic idea behind RDF is to make statements about resources in the form of subject-predicate-object expressions, known as RDF-statements or triples. These RDF-statements can be expressed as a graph by utilizing nodes for representation of subjects, S, and objects, O, where directed edges or arcs represent the predicates, P (Powers, 2003a; Marin, 2006).

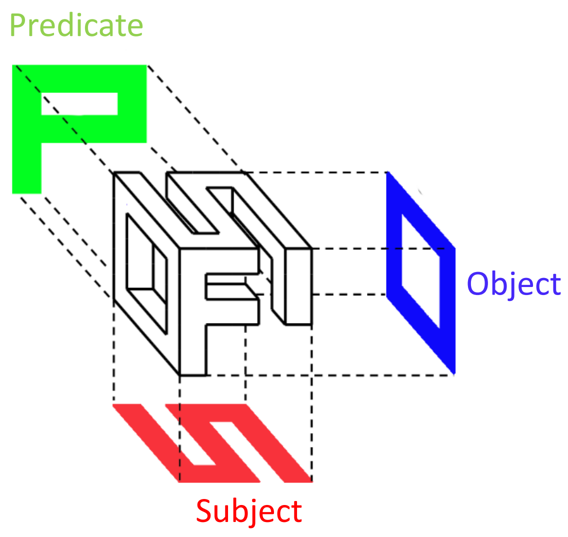


Fig. 2.3: RDF model modified from (Gandon et al., 2011).

An RDF-graph is a combination of triples or RDF-statements in which each triple or statement is a directed graph that consists of a binary property, predicate, p, which connects a subject, s to an object o. Nodes of RDF-statements can either be a URI, a blank node or a literal, whereas an arc can only be a URI.

Marin (2006) in A note on the history of the document: RDF Formalization provided semantic definitions of RDF-graph as following:

- **RDF-URI reference:** An RDF-URI reference is a set of character strings. "Two RDF-URI references are equal if and only if they compare as equal, character by character. υ is set of all RDF-URI references." (Marin, 2006, p. 3).

- **Literal**: "A literal is either a combination of a lexical form ω with a language tag τ to form a plain literal and that is noted as $<\omega, \tau>$ (ω when the language tag is not presented); or it is a combination of a lexical form ω with an RDF-URI reference υ to form a typed literal that is noted as $\omega\ +\upsilon$. Ł is the set of all literals. Literals are distinct from RDF-URI references ($\upsilon \cap$ Ł $= \varnothing$)." (Marin, 2006, p. 3).

- **Blank-nodes:** Blank-nodes are not representing any resources rather than pointing the relations into other resources, blank-nodes are noted as β. "Let β be an arbitrary infinite set disjoint from $\upsilon \cup$ Ł." (Marin, 2006, p. 8).

- **RDF-triples:** "An RDF-triple is an element of $\upsilon \cup \beta \times \upsilon \times \upsilon \cup \beta$ Ł ." (Marin, 2006, p. 7).

- **RDF-graph:** "A set of RDF-triples is called an RDF graph. A subgraph is a subset of triples in the graph." (Marin, 2006, p. 8).

2.5.1.1 Complex n-ary relations

A common approach for representing information is to decompose it into RDF-statements by modeling binary relations between two individuals. In RDF, the property is a binary relation that links two individuals to create a statement (Web, 2015). To Illustrate, consider the following examples.

$$(\overbrace{\text{John}}^{S},\ \overbrace{\text{isA}}^{P},\ \overbrace{\text{Student}}^{O}),\ (\overbrace{\text{John}}^{S},\ \overbrace{\text{hasSon}}^{P},\ \overbrace{\text{Olaf}}^{O}),\ (\overbrace{\text{Olaf}}^{S},\ \overbrace{\text{hasAge}}^{P},\ \overbrace{\ 7\ }^{O})$$

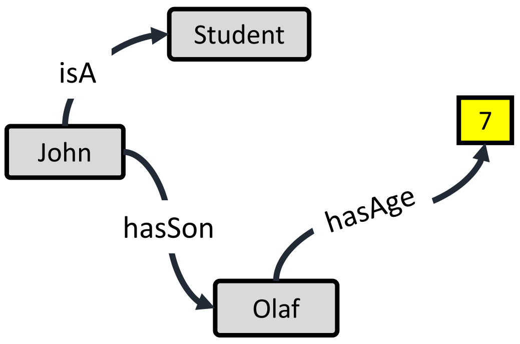


Fig. 2.4: An RDF-graph demonstrating binary relations between nodes

The example mentioned above demonstrates the use of RDF for modeling data. As can be seen in the example shown in Fig.2.4, in RDF an object of a statement can be a subject for another statement.

In most of the cases, RDF-statements are represented by binary relations where a single relation is used to link two individuals or nodes. However, in some cases, this process becomes more complex, and more than two individuals are required to model a single relation. These relations are called RDF n-ary relations. Blank-nodes and reification are two examples of using n-ary relations in RDF.

1. RDF blank-node

   In RDF, a blank-node is a node in an RDF-graph that can represent either a subject or the object of a triple. The blank-node, which is also called an anonymous resource, is not uniquely identified with a URI, and it does not denote any resource othar than referring the relation to other values or resources. The subsequent RDF code is an example of a blank-node in RDF, (Semantics, RDF 1.1, 2014). Blank-nodes are in general used in the following cases:

- Representing multi-valued attributes.
- Security: protection of information from global access.

Listing 2.2: RDF code illustrating the use of blank nodes for modeling RDF data.

```
@prefix rdf:<http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix dc:<http://purl.org/dc/elements/1.1/> .
@prefix ex:<http://example.org/stuff/1.0/>.

<http://www.eti.uni−siegen.de/Modeule732>
            dc:hasName Semantic Web ;
            ex:hasProfessor Kremer ;
            ex:hasStudents _:students.
   _:students:
          Jan;
          Jens;
          Klaus;
          Christian;
```

The above-mentioned RDF example is shown in Fig.2.5.

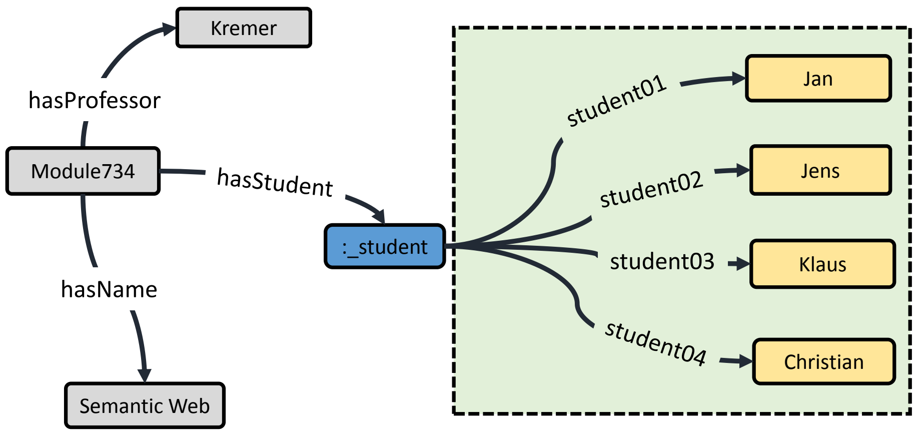


Fig. 2.5: An RDF-graph demonstrating blank nodes

Unlike Internationalized Resource Identifier (IRI)s or literals, blank-nodes, due to a lack of URLs, do not point to concrete real-world objects. Blank-nodes consequently add complexity to RDF. One of the complexities is to detect whether or not two different RDF-graphs are isomorphic. Moreover, checking entailments, leanness and equivalence of RDF graphs are intractable mainly due to the presence of the blank-nodes (Hogan, 2015). The work of (Gutierrez et al., 2004) exposes some drawbacks of using blank-nodes in RDF-graphs; they show that the graph created with the use of blank-nodes is NP-Complete and Isomorphism-Complete. According to Heath and Bizer (2011), the complexity of merging RDF graphs from heterogeneous sources increases when they have utilized blank-nodes; the absence of URI in such nodes might lead to collision, redundancy or increase the size of the resulting graph. To overcome these problems Skolemization[1] has been proposed. The idea is to replace some or if possible all of the blank-nodes with URIs consequently nodes can be identified uniquely. However, replacing all blank-nodes with URIs is complex and raises other disadvantages on its own. For instance, exchanging blank-nodes with URIs will result in creating unnamed elements. To illustrate, consider converting the blank-node which represents the address and refers to the relations to a number of individuals such as street, city, county and so on. As the result, these individuals which should be treated as a group representing the address will be treated as independent individuals, resulting in poor data integration.

Based on its importance, studies have been conducted to construct algorithms which can find an optimal solution in order for the problem of blank-nodes can be traceable. One example is the work of Tzitzikas et al. (2012), where two polynomial algorithms have been proposed to provide the optimal blank-node mapping mechanism. The proposed algorithms have been tested on an RDF graph that contained a large number of blank-nodes; as a result, the proposed algorithms demostrate the performance of mapping 150 thousand blank-nodes per 10 seconds. Another distinguished example is the work of Pichler et al. (2008), where the bounded treewidth concept was introduced for mapping blank-nodes which have a bounded treewidth. Whereas, in the work of Käfer et al. (2013) the process of matching blank nodes was avoided, and alternatively simplistic assumptions were made considering the dynamic of the linked data.

Although blank-nodes are adding complexity to the RDF graph, a considerable number of RDF-graphs are using blank-nodes. A number of studies have demonstrated the advantages of blank nodes for modeling RDF-statements and creating RDF-graphs. One example is the work of Chen et al. (2012), where they demonstrated the functionality of blank-nodes for representing multi-component statements, such as containers, their use for modeling RDF reified statements and also representing complex attributes. Moreover, they demonstrated the advantages of using blank-nodes for protection of sensitive information. According

[1]https://www.w3.org/TR/rdf11-concepts/#section-skolemization

to Hogan et al. (2014, p. 42), "The BTC–2012 corpus – a crawl of 8.4 million RDF documents from the Web – 44.9% of the documents mentioned at least one blank node, 25.9% of the unique RDF terms were blank nodes and 66.2% of pay-level-domains used blank nodes.".

2. Metadata about triples

   Metadata about triples, in general, refers to obtaining additional information about abstract data. It introduces an approach for describing an RDF-statement with another RDF-statement, in other words, it is the process of creating statements about statements (Hartig and Thompson, 2014).

   A statement about a statement, or what can also be called a "meta triple", is, in general, the process of enriching RDF-statements with additional information. For instance, the source, time or place are some information that can be added to any individual statement to provide more semantic information (Nguyen et al., 2014). With the increasing number of resources and information that is collected every day, the need for authentication of information becomes more significant (Almendra and Schwabe, 2006). This authentication can be facilitated by providing provenance information by specifying the origin of the information. In other words, meta-triples can be defined as a process of statement validation by means of another statement (Buneman et al., 2006; Powers, 2003b).

   Representing a meta-triple is an essential requirement for modeling RDFs, however, this is not a straightforward process. Creating such triples is a challenging process that encounters another level of complexity. Looking back into the history of the SW, in 2004, reification became one of the W3C recommendations for meta-triple representation. However, due to its limitations, it has been withdrawn from the latest recommendation of the W3C organization and has become non-standard (Hayes, Patrick, 2004; Nguyen et al., 2015).

   The importance of this concept, however, brought attention to a large number of researchers, several studies have been conducted, to address the problem of meta-triple representation (Groth et al., 2010; Sahoo et al., 2011; Callahan and Dumontier, 2013; Nguyen and Siberski, 2013). A number of approaches have been proposed to provide an optimal representation of meta-triples in the RDFmodel. These approaches can be furthermore classified into the following groups:

   - Quintuple

    - RDF +: In RDF+, each triple is annotated with an internal identifier to be used for the assertion of the metadata. In this approach, a formal semantic and an abstract syntax creates mappings of RDF+ to the RDF, and vice-versa, it also is extending the semantic query language to support RDF+ triples. Consequently, it is inconsistent with the W3C standards.

- Quadruple

    - Named graph: In this approach, a fourth element is added to the graph and represents its provenance. The named graph is used to annotate a triple and also provide a unique identifier to it as shown in Fig.2.6. This approach, however, is not intuitive and it is representing a departure from its original proposed application (Carroll et al., 2005).

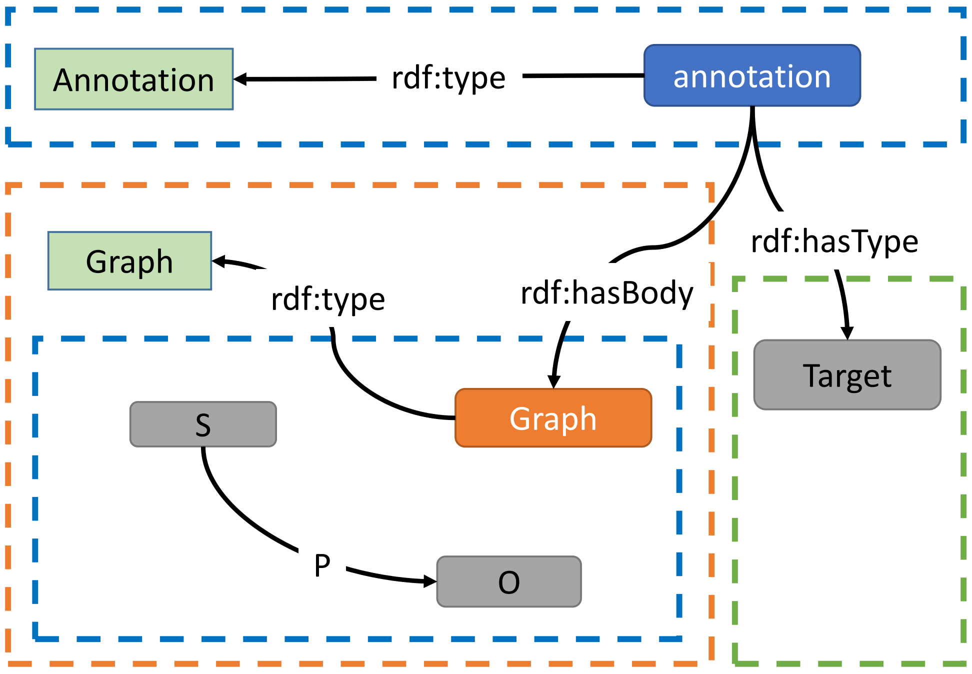


Fig. 2.6: An RDF-graph demonstrating the Named Graph approach (modified from (Semantic Annotation, 2013))

- Triple

    - Reification: In reification, a triple can be reified by another statement, it is also known as statements about statements, it is the process of enriching an RDF-statement with another one (Hartig and Thompson, 2014; Semantics RDF 1.1, 2004).

Listing 2.3: RDF-statements illustrating the use of reification for modeling statements about statements in RDF.

```
"To simplicity, prefixes are omitted in this example."

st1: rdf:Statement;
           rdf:subject: Apple;
           rdf:predicate: hasTase;
           rdf:object: Sweet

 rdf:subject: Klaus;
 rdf:predicate: Says;
 rdf:object: st1;
```

The above mentioned example is demonstrated in an RDF-graph and shown in Fig.2.7.

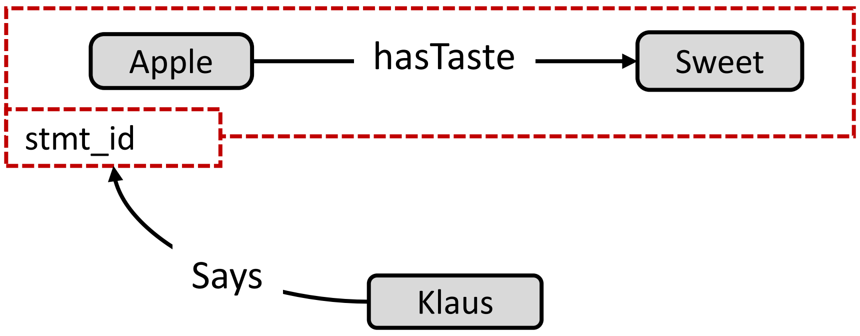


Fig. 2.7: An RDF-graph demonstrating the reification approach

However, in this method, as the metadata information is not bounded to the triple the reasoning about reification is not possible (Semantics RDF 1.1, 2004; Nguyen et al., 2015).

– Singleton Property (SP): This approach has intended to address some of the problems of RDF+. The approach suggests creating a SP for every context in the graph to uniquely identify triples, and then to use them for the assertion of metadata to these triples. The advantages of the SP approach over RDF+ is that in SP the metadata is represented in the form of triples which provide the compatibility to the W3C standard for the RDF model as well as its query language. However, SP on the contextualized predicates, which lead to the generating of a large number of properties, is a challenge for reasoning as well as querying such properties (Nguyen et al., 2014; Nguyen et al., 2015).

2.5.1.2 Simple protocol and RDF query Language

Along with the release of RDF as a W3C recommendation in 1998, the problem of querying such data models was raised. To overcome this problem, several languages for querying RDF triples have been proposed (Pérez et al., 2009).

Haase et al. (2004) in their work address the problem of querying RDF and implement a precise comparison of five different query languages for RDF. In their comparison, along with eliciting the compatibility to the RDF data model, the following criteria have been considered for each language:

- Expressiveness: It is the power of a query language to form queries in a given language, for instance, its ability to implement relational algebra.
- Closure: It is the means of ensuring that the result of any query is with the boundary of the data model and it is again an element of the same data model.
- Adequacy: It is the ability of the query language to use every concept of the data model.
- Orthogonality: It is the ability of the query language to use all operations independently of the language context.
- Safety: It is the ability to return a finite set of outcomes for all syntactically correct queries.

According to the criteria mentioned above, Haase et al. (2004) compared six different query languages for their compatibility and functionality for querying RDF-triples, the compared languages were, RDF Query Language (RQL), Sesame RDF Query Language (SeRQL), Versa, Notation3 (N3), and RDF Data Query Language (RDQL). The researcher conducted a number of use cases by testing queries that could be expressed. If the given query could be formulated, a half of a point was given to the query language. Considering this metric, all five query languages were compared. The results show that RQL and SeRQL appear to be the most complete languages among others, they were able to answer most of the queries and collected 10.5 and 8.5 out of 14 points. Versa collected 7.5 points followed by N3 with 7 points. Triple and RDQL with 5.5 and 4.5 points were at the end of the list. However, none of the mentioned query languages were able to fulfill all desired requirements to be a standard RDF query language.

Another example is the work of Furche et al. (2006) where comparisons and a time-line development of RDF query language have been presented. For their study Furche et al. categorized the selected query languages into three main categories as follows:

1. "Relational or pattern-based query languages", RQL, TRIPLE and Simple Protocol and RDF Query Language (SPARQL).

2. “Reactive rule query language", Algae.

3. "Navigational access query language", Versa

In the work of Furche et al. (2006) all above-mentioned query languages have been studied and compared in depth. Moreover, a time-line showing the development of RDF query languages has been provided as shown in Fig.2.8. Furthermore, the authors have mentioned advantages of the SPARQL over other query languages and stated, "SPARQL is a query language that has already reached candidate recommendation status at the W3C, and is on a good way to become the W3C recommendation for RDF querying" (Furche et al., 2006, p. 4).

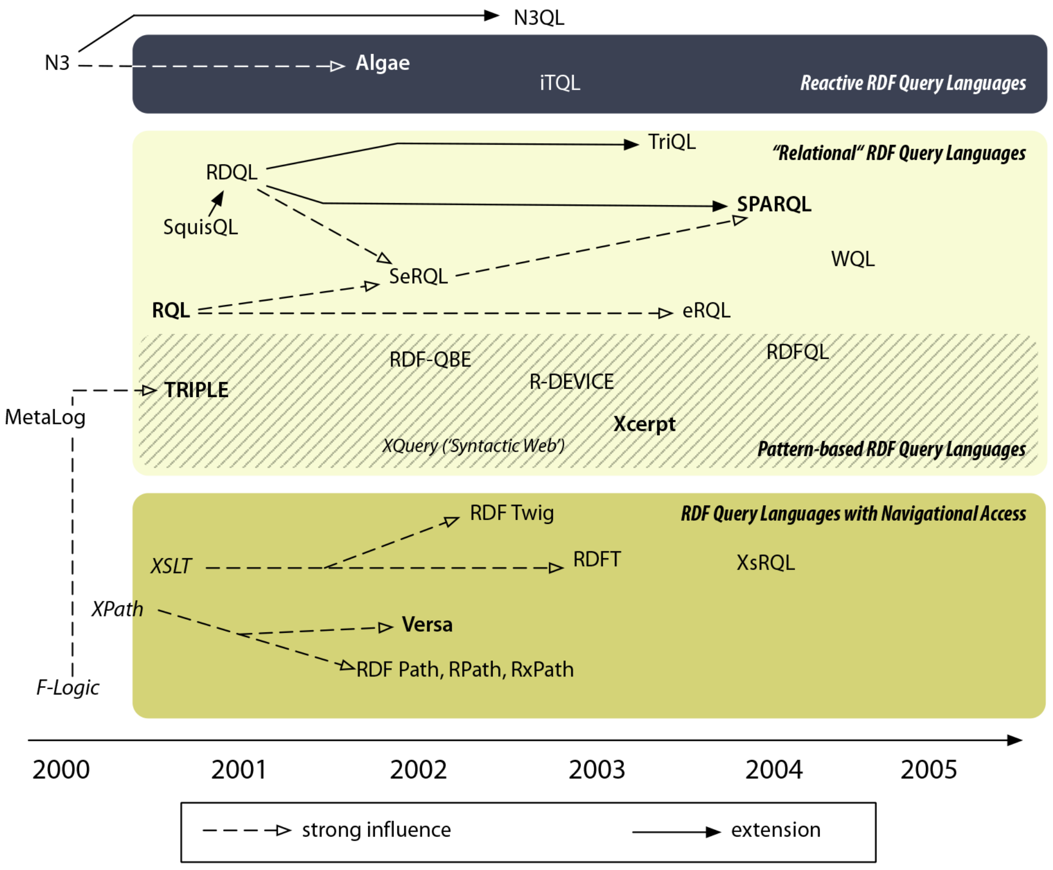


Fig. 2.8: The time-line of query language development according to (Furche et al., 2006).

In 2008, SPARQL 1.0 became the standard query language for RDF and the W3C recommendation (Herman, Ivan, 2008).

RDF is a directed graph, and, therefore querying an RDF graph with SPARQL which is a graph matching query language, amounts to matching graph patterns that are presented as a set of "subject, predicate, objects" triples (Furche et al., 2006; Pérez et al., 2009). In SPARQL the query that is represented by Q defines a graph pattern P to match against G which is an RDF graph, an example is shown in Fig.2.9.

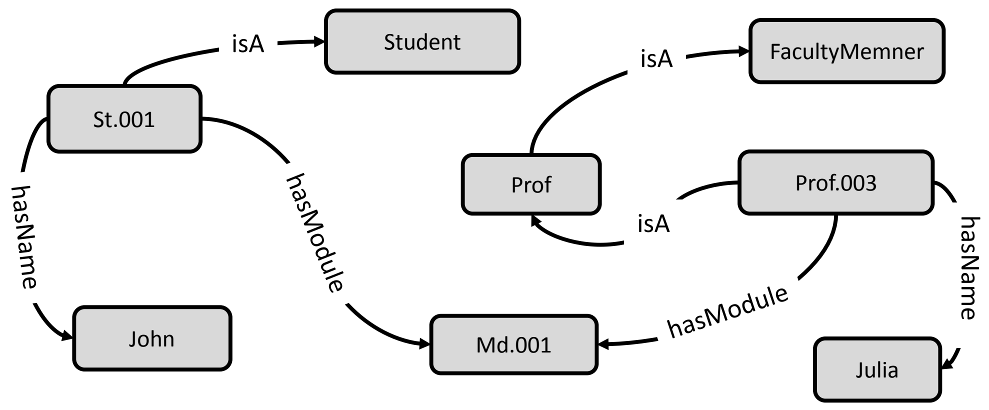


Fig. 2.9: An example of RDF graph pattern G that can be queried with the SPARQL.

To achieve pattern matching, SPARQL replaces elements of P with elements of G. The SPARQL's basic notion is called triple pattern tp (Schätzle et al., 2016).

$$tp = (s', p', o'), \quad with \quad s' \in (s, ?s), \quad p' \in (p, ?p), \quad o' \in (o, ?o)$$

Elements of triple pattern are either RDF terms or so-called *bound* variables which are indicated by a question mark *?* and are called *unbound*. A set of triples pattern will form a basic graph pattern called *BGP* as shown below (Schätzle et al., 2016).

Listing 2.4: A example illustrating the use of SPARQL for querying RDF-statements.

```
SELECT * WHERE {
              ?subject hasName ?name.
              ?subject hasModule ?module.
              ?subject isA ?object.
           }
```

One advantage and application of the SPARQL is its ability of querying over an RDF graph with blank-nodes. A blank-node in a graph pattern has the following characteristics:

- Due to a lack of a URL, it is a non-distinguishable variable.
- A blank-node cannot appear in the selected clause.
- It can be represented by "**_:b**" or the abbreviated form **[]**.

An example of an RDF graph is shown in Fig.2.10

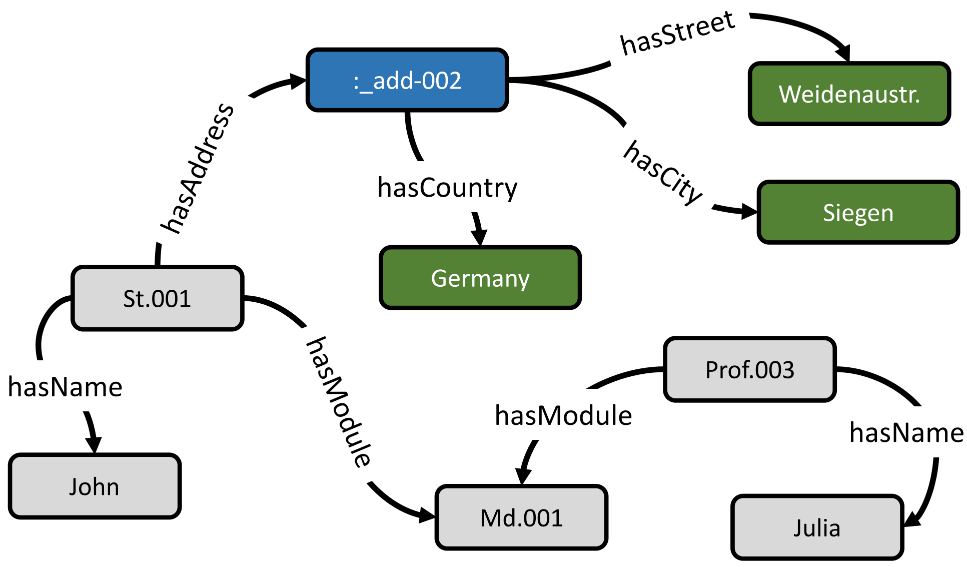


Fig. 2.10: An example of a blank-node application in RDF graph pattern.

The subsequent SPARQL code demonstrates the functionality of SPARQL for querying RDF graphs containing blank-nodes.

Listing 2.5: An example is illustrating the use of SPARQL for querying over blank-nodes.

```
SELECT * WHERE {
                ?s hasName ?o.
                ?o hasModule ?g.
                ?o isA ?k.
                [] hasAddress ?address
             }
```

2.5.2 Ontology modeling

Let us start this section with one of the most profound paragraph and the most quoted ontology definitions of Gruber.

"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are presumed to exist in some

> area of interest and the relationships that hold them (Genesereth and Nilsson, 1987). A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly. **An ontology is an explicit specification of a conceptualization**. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what “exists” is exactly that which can be represented" (Gruber, 1993, p. 200).

The source for the concept of ontology can be traced back to ancient philosophy, where the Greek philosopher Plato (429–347 BC) proposed answers to some of the significant questions that arise throughout modeling: "*What is reality? What is existence? What is the true nature of things?*". Plato noted the first contribution to the conceptualization of ontology (Hitzler et al., 2009). The reflection of ontologies into information technology was first noted in 1997 by Vickery. Hence, the library and information science (LIS) community have endured an enormous improvement in the related areas of ontology research, mainly in response to the W3C organization and the SW vision of Berners-Lee (2001). Ontology has become important, particularly in semantic technologies and in the field of knowledge representation, ontology can be defined as a knowledge description of a domain of interest in a machine-processable form (Hitzler et al., 2009).

The SW intends to utilize and bring together philosophy, information systems, AI and SW tools to formalize a standard recommendation for ontology development. In the year 2004, the W3C announced some standards and recommendations for ontology development and for facilitating the information exchange over the web (Hitzler et al., 2009). These standards are the Resource Description Framework Schema (RDFS) and Ontology Web Language (OWL).

2.5.2.1 Resource Description Framework Schema, RDFS

Section 2.5.1 describes how a statement about a single resource can be formed in RDF. Basically, RDF uses three elements to describe a single resource. In other words, three individuals (a subject, a predicate, and an object) are in relation to create an RDF-statement. However, when defining a new domain of interest, it is necessary to use terms that are not only representing the individuals and their relations but moreover, can be used for types or classes. A collection of such terms are called vocabularies. Vocabularies are terms that can be used to represent individuals, relations, and classes. (Hitzler et al., 2009).

RDFS is an extension of RDF, which is part of the W3C recommendation, it is aimed at defining vocabularies. RDFS is a lightweight semantic language used for declaring and describing classes and properties. Classes in RDFS referred to resources types and properties to attribute types.

A class can be defined as a set of elements sharing the same properties. Instances of a class is a term used for individuals belonging to a specific class, the process of adding individuals to a class is referred to as instantiating a class. In RDF, an *rdf:type* is used to define the relationship between an instance and a class. Classes are related to each other, thus, it is essential to establish relationships between them. One of the most important kinds of relations is the class hierarchy. For instance, class A is a subclass of class B if all instances of class A are also instances of class B.

In RDFS, due to the global properties definition, new properties can be defined and applied to existing classes without the need of changing classes. Although this method of handling properties deviates from the standard approach of modeling and object-oriented programming, it is still a powerful mechanism for providing the capability of reusing the classes which are defined in other domains and adapt them into our requirement. These same principles can be applied to RDFS properties. (Antoniou and Harmelen, 2004).

The following example will demonstrate the different layers of RDF and RDFS. To illustrate, consider the following RDF statement:
John studies module 001.
The RDFS schema for the above mentioned statement may contain the following classes, person, male, female, student and course. As for properties, it may contain *involves* and *hasModule* , Fig.2.11 shows different layers of RDFS and RDF for modeling and instantiating an ontology.

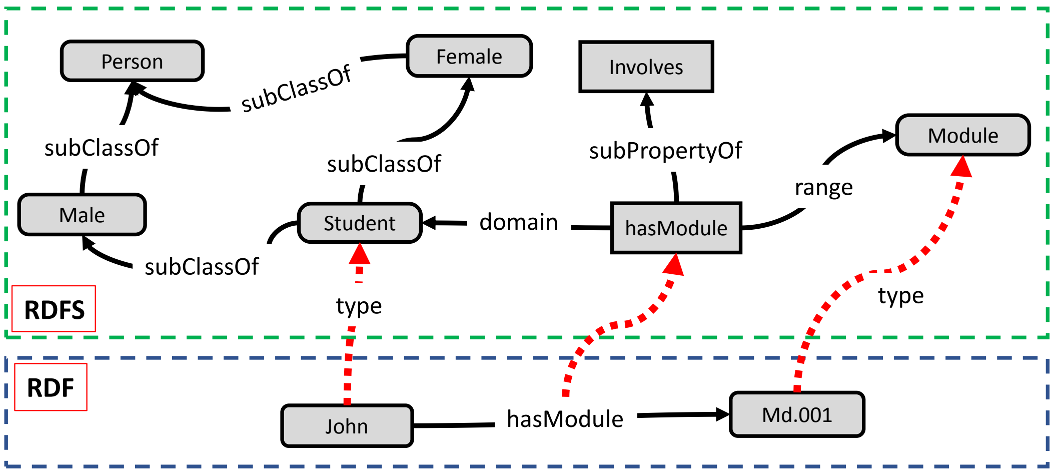


Fig. 2.11: An example demonstrating the different layers of RDFS and RDF.

As mentioned earlier, RDFS is used to define vocabularies by declaring classes and properties, their hierarchies, relationships, and their domains and ranges restrictions. However, RDFS encounters a number of limitations for modeling sophisticated ontologies (Antoniou and Harmelen, 2009). Below are some examples of RDFS inability to provide the following features:

- Disjoint classes: In an ontology, two classes are disjointed if they are not able to share an instance. For example, male and females are both subclasses of person but they are also disjoint classes.

- Boolean operations among classes: The ability of an ontology to build new classes through a combination of existing classes using complement, intersection, and union operations.

- Cardinality restrictions: The ability to place restrictions on ontology by defining the properties values. For instance, a person hasOnlyOne mother.

Despite its usefulness, due to the above-mentioned limitations RDFS has been classified as an ontology lightweight language. Consequently, a more expressive language is required to model ontologies for more sophisticated applications.

2.5.2.2 Ontology Web Language, OWL

As described, RDFS is not suitable for modeling complex ontologies. For modeling such complex ontologies a more expressive language based on formal logic is required. Formal logic enables us to access the implicitly modeled information from the ontology, and OWL is considered as such a language.

OWL is a language designed for processing and integrating the information to create ontologies. OWL is built over RDFS and became a W3C standard in the year 2004. OWL, in general, utilizes the characteristics of RDF and RDFS and extend their functionality to another level. Just like RDFS, OWL defines classes, subclasses, and their hierarchies. Moreover, it provides functionalities beyond RDFS capabilities, such as creating classes from unions, intersections, and complements of other classes. The ability to define the disjoined classes, equivalence statements on properties, and inequality and equality between individuals are other advantages of OWL. In OWL, classes can be defined with restricted properties to ensure that the value of the given properties belongs to a particular class (Horrocks et al., 2003; Hitzler et al., 2009).

As described, OWL can be considered as a sophisticated and robust language for modeling ontologies. It has recently seen a rapid increase in its popularity, its presence in SW, AI, the IoT and robotics is undeniable. After all, OWL is an expressive language and provides flexibility of choice between different levels of expressivity, as shown in Fig.2.12.

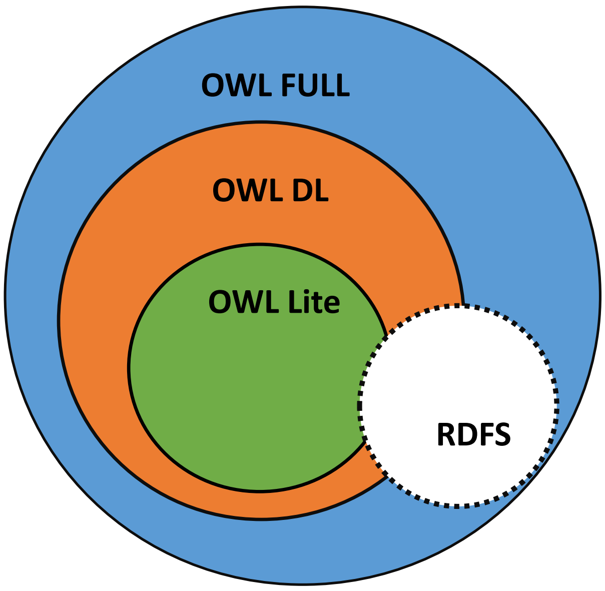


Fig. 2.12: Illustration of the sub-languages of OWL, modified from (Bergman, 2018).

OWL introduced the following three sub-languages:

- OWL Full: OWL Full provides the flexibility of using all OWL primitives as well as allowing the combination of its primitives with RDF and RDFS. It contains both OWL DL and Lite as well as RDFS, and it is fully compatible with RDF. However, OWL Full is very expressive, and therefore for any software tool it is hard to be used, and it refrains from full reasoning support (Antoniou and Harmelen, 2009; Hitzler et al., 2009).

- OWL Description Logic (DL): OWL DL is a sub-language of OWL Full, and it contains OWL Lite. To provide computational efficiency and reasoning capability it restrains from using a number of RDF and RDFS contractors. Therefore, it is widely supported by software tools (Antoniou and Harmelen, 2009; Hitzler et al., 2009).

- OWL Lite: OWL Lite is a sub-language of OWL DL, it has the most restriction and limitation among others. For instance, in OWL Lite, disjointness statements, enumerated classes, and arbitrary cardinalities are omitted. Therefore, this language is less expressive, and it is much easier to implemented (Antoniou and Harmelen, 2009).

When developing ontologies, depending on the desired ontology an OWL sub-language should be considered. The choice between Lite and DL mainly depends on the level of expressiveness, while the choice between DL and FULL mainly depends on the extent to which the RDFS facilities are required (Antoniou and Harmelen, 2009).

The compatibility between OWL sub-languages are as follows:

- All valid Lite conclusions are also valid DL conclusions.
- All legal Lite ontologies are also legal DL ontologies.
- All valid DL conclusions are also valid Full conclusions.
- All legal DL ontologies are also legal Full ontologies.

In their recent update, the W3C organization has announced OWL2. In this new version, as shown in Fig.2.13, the sub-language Lite has been omitted and, instead, three new sub-languages have been added.

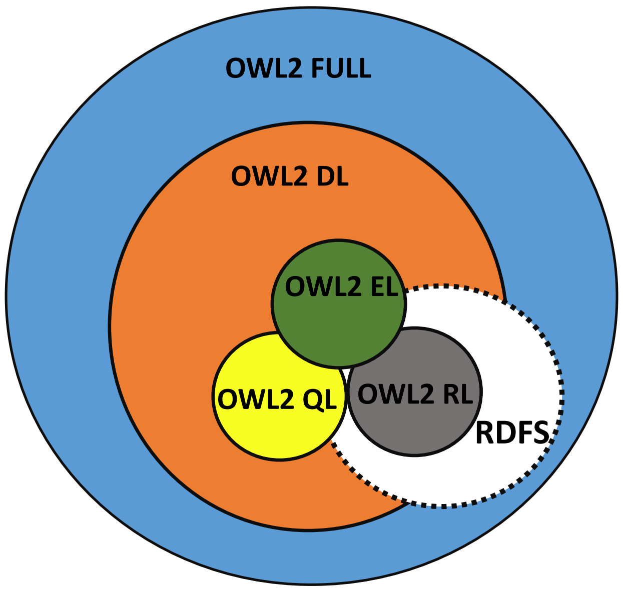


Fig. 2.13: Different layers of OWL2, modified from (Bergman, 2018).

OWL2 introduces the following sub-languages:

- OWL Rule Language (RL): OWL2 RL enables the polynomial time algorithms to be implemented with rule-based reasoning. It is designed to simplify the adaptations of OWL by rule-based inference tools.

- OWL Query Language (QL): OWL2 QL enables the implementations of relational database systems for conjunctive query answering. Moreover, it features polynomial time algorithms. The OWL2 QL is designed for data-driven applications (Hitzler et al., 2009).

- OWL Entity Language (EL): OWL 2 EL enables polynomial time algorithms, such as instance checking satisfiability checking and classification. It is designed for modeling ontologies with a vast number of classes and role hierarchies and only a limited number of OWL features (Hitzler et al., 2009).

These new lightweight sub-languages are designed to simplify the reasoning over ontologies. These improvements lead to an increase in the performance and the scalability of OWL (Krötzsch, 2012).

2.6 Semantic knowledge acquisition from sensory data

In the beginning the Internet was characterized by a network of linked static HTML documents. When Web 2.0 was introduced, the two-way communication became possible; Web 2.0 technologies introduced the social networking services, such as blogs and soon become indispensable to our society where global marketing, trade and also social interaction over the Internet were possible. Web 2.0 however with all of its advancement has limited sources and structure for machines to communicate and interact with each other (Whitmore et al., 2015).

The goal of SW or what is so-called Web 3.0 is to provide an environment where machines can process, understand and interact with each other without human involvement. On the other hand, alongside Internet development, the development of sensors has also been also evolving; smaller, more sensitive and smarter sensors are dominating our world. The convergence of these two great technologies can lead our world into a new era where machines can use the Internet and interact and process data instead of human beings. This is a new vision and possibility for our future, this vision has created the IoT. The IoT can be referred to as a platform in which sensors have the capability to interact and communicate with each other to achieve their objectives (Whitmore et al., 2015).

The IoT is providing to numerous types of sensors the possibility of integrating the real-world data into Internet; it is a framework in which billions of sensors interchange real-world data. This data gathering is facilitated by a vast number of sources such as sensors of computers, smartphones or sensors of smart cities (Ganz et al., 2016).

Nowadays, around the world, billions of sensors in smartphones along with millions of other sensors, which are being deployed in our cities, are connected and are collecting real-world data within the Internet (Gyrard et al., 2017). However, managing a huge amount of sensory data is not straightforward but rather a challenging task. According to Compton et al. and Tollefson (2011), the process of making sense and managing sensory data is an ongoing challenge (Stocker et al., 2012).

One approach to manage and processing data in a machine inter-operable format is using semantic technology. Specifically, ontologies are employed to present this information in a machine-readable form. A considerable number of ontologies have been created to represent real-world data, yet a small effort has been made in automatic real-time ontology population or instantiation from sensory data of mobile robots (Ganz et al., 2016).

In the work of Dietze and Domingue (2009), the importance as well as challenges of using semantic technology for representing sensor data have been presented. The authors stated that the vast growth of sensory data increases the demand for modeling them semantically with a formal language such as RDF and OWL. This is a challenging process, for instance, data collected from sensors are usually relying on measurements of perceptual characteristics whereas in ontologies the data are represented through symbols and are easier to interpret.

Another example is the work of Stocker et al. (2012), where machine learning has been used to implement a system in which the sensory data of a vibration sensor could detect and classify road vehicles which are passing a street,and this data is then used to create an ontology. The work, however, was not fully automatic, it required human interaction and was dependent on domain training for a specific environment.

Recently a study conducted to use the sensor data of a single sensor platform to detect the weather change measurements for automatic ontology creation. In this work, a rule-based framework has been introduced that uses the statistics as well as, k-means clustering methods. The constructed ontology from the proposed framework can be used for monitoring weather-cast applications (Ganz et al., 2016).

2.7 Ontologies and robotics

In general, interaction and communication are two of the essential elements of any intelligent robot. Robot communication with a human, on the one hand, eases its entrance into the human's life, and on the other hand, its communication with other robots raise its capability in multi-robot applications. This communication need raises the need for a formal and standard knowledge representation. To this end, ontologies can play an essential role in enabling semantic and reliable data integration and also information exchange between robots, and it can be used to express information in an understandable human way (Schlenoff et al., 2012; Carbonera et al., 2013).

In 2011, the Robotics and Automation Association (RAS) recognized the need for ontologies in robotics and consequently created a working group called the Ontologies for Robotics and Automation (ORA). The goal of the ORA was defined as developing a standard for providing an ontology with its underlying methodologies for knowledge representation in robotics. The ORA group introduced the IEEE standard ontologies for robotics and automation. This ontology composed the Core Ontologies for Robotics and Automation (CORA) which was designed to define a robot and its relation to other concepts. The ontology has been developed over Suggested Upper Merged Ontology (SUMO) ontology, which is an upper ontology, and is considered a foundation ontology for CORA. CORA defines class hierarchy and its related relations and rules. According to SUMO all entities are divided into either physical or abstract entities, an overview of basic concepts of the ORA is shown in Fig.2.14.

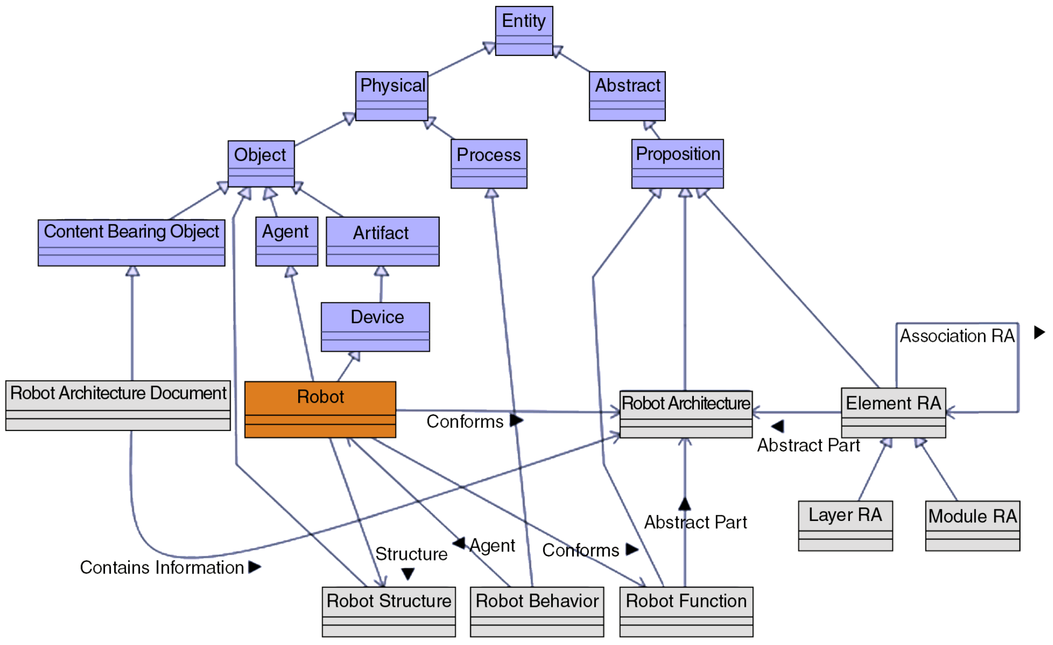


Fig. 2.14: The basic concepts and relations introduced by the ORA (Fiorini et al., 2017).

In general, the ORA defines robots, its parts, robotic systems, and complex robots. This standard is considered as a reference point for ontology creation and knowledge representation in robotics (Fiorini et al., 2017).

Due to its importance different studies have been conducted and a considerable number of domain ontologies have been proposed for robotics. According to Kuipers et al. (2000) the robotics domain is classified into subsequent levels, metrical level, sensory level, control level, topological level (classes of paths, regions, and places), and causal level (action, views, events and their inter-relations), and each of the mentioned levels have been further enhanced by their sub-ontologies. Whereas Paull et al. (2012) categorized robotics domain into sensors and actuators, power, control algorithms, kinematics, locomotion, dynamics, localization, communication, and planning (Plauska, 2013).

According to their domain level, different ontology domains have been introduced for semantic knowledge representation in robotics. In the works of Plauska (2013) and Compton et al. (2009b), surveys and compressions of different robotics ontology domains have been presented; in the Table 2.1, the most common ones are outlined.

Ontology	Domain Level	Application
Plateforme pour la Robotique Organisant les Transferts Entre Utilisateurs et Scientifiques (PROTEUS)	Environment, robot	Ontology designed for modeling missions of a mobile robot.
Robot description ontology	Robot features, environment, and its conditions and emergencies	Ontology created to describe robot's feature as well as complex search and rescue scenarios.
Spatial Ontology	Location regions, topologies and spatial relations	An ontology designed especially for navigation.
Agent-based Middleware Approach for Mixed Mode Environments (A3ME)	Sensor, Devices, actuator, energy and computing	Ontology for device discovery heterogeneous network source.
OntoSensor	Sensor measurement and capabilities	An ontology created for technical specs of sensors.
Semantic Sensor Network (SSN)	Sensors and actuators.	An ontology designed to describe sensors and actuators and their properties, and capabilities.
OntoSTIT	Actions	An ontology created to represent various types of actions.
WIreless Sensor Networks Ontology (WISNO)	Wireless sensors and their data types	An ontology designed for describing wireless sensors and their properties, and data types.

Tab. 2.1: Robotic ontologies according to their domain level.

The diversity of work indicates the importance of ontologies in robotics, however, for a mobile robot navigating in a dynamic unknown environment, the use of pre-defined ontologies is not sufficient. The robot should be able to create or instantiate its ontology regularly with new information to be able to adapt to the changes in the environment; this is an ongoing challenge and open problem in the field of robotics.

2.8 Ontologies and question answering systems

Question answering (QA) system can be defined as a discipline in the fields of natural language processing (NLP) and information retrieval. NLP itself is a branch of AI that defines techniques and methods for the automatic processing of natural languages. NLP attempts to capture natural language and process it using rules and algorithms. It uses different methods and results from linguistics and combines them with modern computer science and artificial intelligence.The goal is to create the most direct and extensive communication possible between humans and machines (Litzel, 2018). QA, on the other hand, defines techniques and methods of intelligent agents for answering human questions in a natural language. QA can be referred to as the process of translating questions asked by users and mapping them into their semantics with natural language representation and providing a valid answer accordingly (Trischler et al., 2016).

In QA the machine needs to work beyond keyword matching, it should furthermore understand the semantics of the question and, based on that, find the proper answer without human aid. According to the work presented in (Trischler et al., 2016), QA can be classified into the following three main paradigms:

- Information Retrieval (IR)-based Factoid QA approach: In this approach, the aim is to provide each question with a short segment answer. Hence, in the question processing phase, question statements are divided into pieces of information, and answer types are identified. Consequently, the IR system is provided with queries which specify the search keywords to be used for documents in the web or other datasets.

- Knowledge-based QA approach: In this approach, the questions are answered based on structured databases for instance relational or ontological such as RDF databases. Here, question statements are mapped into semantic phrases which can be queried with the help of query languages like Structured Query Language (SQL) or SPARQL.

- Hybrid QA approach: In this approach, the questions pass through different phases in order to find the best suitable answer. First, in the question processing phase, QA runs relation extracting, parsing and the named entity tagging on the question. The question is then classified based on its focus and answer type. It is later combined with additional information and documents to recommend candidate answers. In this approach, these candidate answers can be extracted either from structured databases or documents. Finally, the candidate answers are validated, scored and ranked. The highest scored candidate answer is then considered as the answer to the given question.

Due to the importance of QA several studies have been conducted and a number of systems have been proposed, each implementing different approaches. For instance, in the work of Schlaefer et al. (2006), where the OpenEphyra was introduced, the proposed system was, however, only able to answer a set of basic and fixed questions. Another example is the work of Ferrucci et al. (2009), where Open Advancement of Question Answering (OAQA) was introduced. Nevertheless this approach was incomplete, considering it did not provide an end-to-end pipeline. A more precise approach has been proposed in the work of Marx et al. (2014), where OpenQA was introduced, OpenQA provides an end-to-end pipeline platform, however, it cannot process unstructured data.

International Business Machines (IBM) also recognized the importance of QA systems and created a research group from twenty-one researchers to address the challenges of QA. In 2011, and after three years of research and development, the group introduced a DeepQA framework through which the IBM Watson question answerings system was developed. The system contains hundreds of complex algorithm to evaluate the answering process in different phases as shown in Fig.2.15. In DeepQA different information is extracted from each question statement, processed and enriched with additional information such as focus types. The system utilizes NLP technology throughout the process for interpreting questions and extracting its key element and semantic. In the answering phase, the number of candidate answers are provided which are later evaluated and scored. The highest scored candidate answer is then considered as the answer (Lally, 2011).

Although IBM Watson could successfully process the unstructured data, it has some limitations when it comes to the structure data and more precisely ontologies, it was not able to process RDF data. Therefore Watson was not suitable for SW application and ontology source knowledge-bases.

To overcome the limitations of the previous QA systems, Baudiš and Jan Šedivý (2015) introduced a new QA system, named Yet anOther Deep Answering pipeline Question

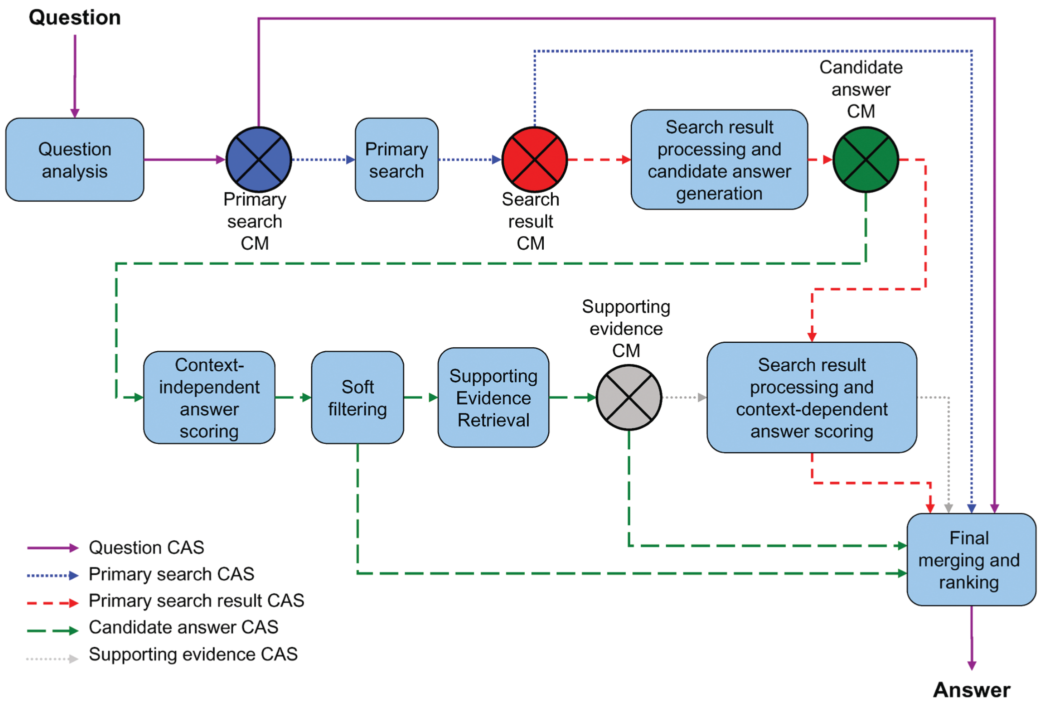

Question
Question analysis
Primary search CM
Primary search
Search result CM
Search result processing and candidate answer generation
Candidate answer CM
Supporting evidence CM
Context-independent answer scoring
Soft filtering
Supporting Evidence Retrieval
Search result processing and context-dependent answer scoring
Final merging and ranking
Question CAS
Primary search CAS
Primary search result CAS
Candidate answer CAS
Supporting evidence CAS
Answer

Fig. 2.15: The DeepQA architecture (Epstein et al., 2012).

Answering (YodaQA). YodaQA was introduced with the intention of providing a modular and open source end-to-end pipeline system, which allows the integration of different knowledge bases. YodaQA is inspired by the IBM Watson framework using DeepQA, and its pipelines are implemented using the Apache Unstructured Information Management Architecture (UIMA) framework (Baudiš and Jan Šedivý, 2015). The process of answering questions in YodaQA is passing through the following phases as shown in Fig.2.16.

1. Question reader: This the initial input phase. In this phase, the collection reader acquires question(s).

2. Question analysis: In this phase the system utilizes the NLP for extracting features from the question statements from the previous phase, these features are :

   - Clue: refers to the key information on the question statement. For instance in the following question "where is the Siegen university?" the clues are:"Siegen", "university", and "Siegen university".

   - Selective Verb (SV): refers to the coordinating verb of the question statement.

   - Focus: refers to the center point of the question statement.

- Lexical Answer Type (LAT): refers to the type of answer that would fit the question.

- Subject: refers to the main "topic" of the question statement.

3. Answer production: In this phase, based on the question analysis phase, a set of candidate answers are created.

4. Answer analysis: In this phase, the answer candidates of the previous phase are validated based on various answer features, and each candidate is then attached with a score.

5. Answer writer: In this phase, candidate answers, their scores, and origin are delivered to the user.

YodaQA is a promising QA approach which is suitable for both structured and unstructured data sources. With a correct answering rate of about 80%, it is considered as one of the most accurate QA systems available.

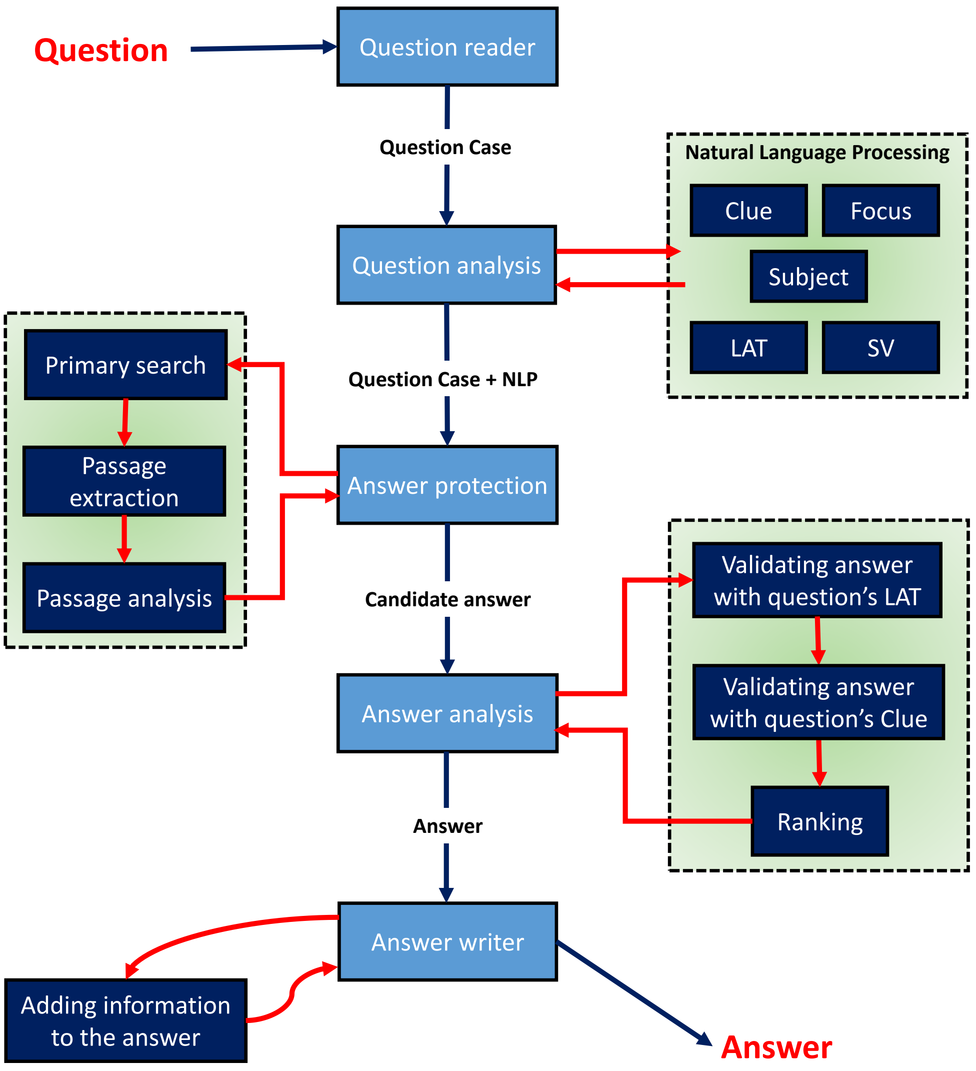


Fig. 2.16: The YodaQA architecture

3 Technical setting of the proposed system

> *We must ensure that technology is accessible, affordable, and adds value.*
>
> — **Narendra Modi**
> (the prime minister of India)

3.1 The Autonomous Mobile Outdoor Robot, AMOR

AMOR (short for Autonomous Mobile Outdoor Robot) is an autonomous outdoor mobile robot from the Institute for Real-Time Learning Systems at the university of Siegen. It is based on a Yamaha Kodiak 400 quad equipped with various actuators and sensors, such as, a Global Positioning System (GPS), laser scanners, weather station, imaging camera, as well as, Ultrasonic sensors to capture its condition and its surroundings (Kuhnert, 2008).

AMOR combines off-road capability, high range and a robustness against weather and environmental influences. A central point of AMOR is the use of low-cost sensors that can exploit their respective strengths in different weather and environmental conditions. The first significant success of AMOR was it's victory of the Urban/Non-urban Autonomous Reconnaissance Scenario at C-ELROB 2007 (Kuhnert, 2008).

It is worth mentioning that any mobile platform can be employed by the RoboSemProc, however, in this work the AMOR has been addressed.

3.2 AMOR in simulation

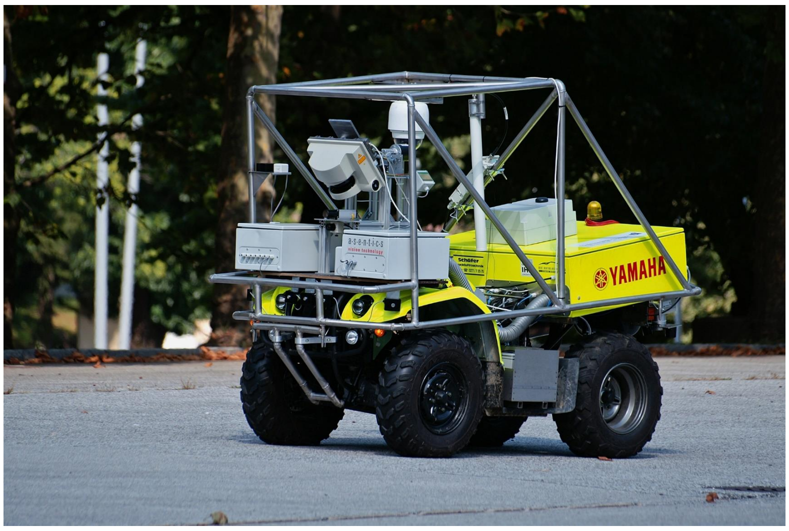


Fig. 3.1: AMOR navigating in an outdoor environment.

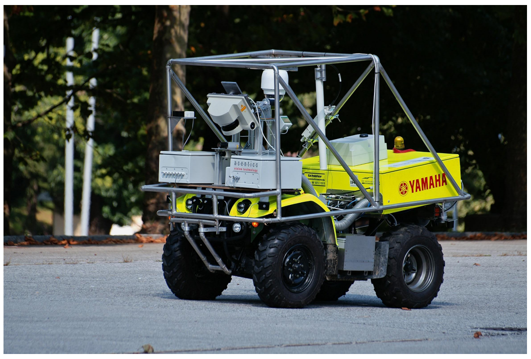
3.2.1 AMOR in MORSE environment

Based on the advantages of the simulation described in section 2.4, and particularly in the following works (Shannon, 1998; Kehoe et al., 2015; Duarte et al., 2014; Qu and Yuan, 2015; Hanna and Stone, 2017; Roehrbein et al., 2016; Robotic Industries Association, 2018), the simulation technique has been used to ease the process of navigation and data collection. Therefore, the simulated model of AMOR has been employed. The simulated model of AMOR was presented in this work as a work of the Institute of Real-Time Learning Systems. The simulated and real model of AMOR can be seen in Fig.3.2b and Fig.3.2a accordingly.

The robotic modulator, Modular OpenRobots Simulation Engine (MORSE)[1] is a generic simulator for academic robotics. MORSE provides a number of simulated sensors such as different cameras, laser scanners, GPS and odometry sensors; it also comes with a set of actuators such as high-level waypoints controllers, speed controllers, etc. MORSE, moreover, has its own set of robotic models, but it also provides the ability to add new robot models (Openrobot, 2009). The simulated model of AMOR has been added to the MORSE environment.

The simulated AMOR has been equipped with the following sensors and actuators:

[1]https://www.openrobots.org/morse/doc/stable/morse.html



(a) The real model of AMOR

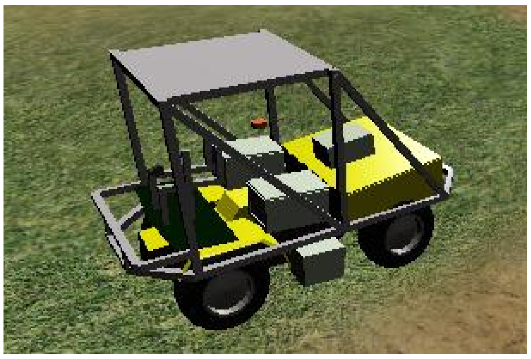

(b) The simulated model of AMOR

Fig. 3.2: The simulated and real models of AMOR.

- **Sensors**
  - **Pose**[2]: As the name indicates the pose sensor returns the full pose of the sensor; it provides the position as well as the orientation of the sensor in the MORSE environment. The pose sensor exports the following data fields with the rate of 60 tics per second:
    - ⋆ **Timestamp**: Returns the time past in simulation in seconds.
    - ⋆ **Position**: Outputs the x, y and z coordinate positions of the sensor, showing its exact position in the simulated MORSE environment.
    - ⋆ **Orientation**: Provides the yaw, pitch and roll, showing the sensors rotation around the Z-axis, Y-axis and X-axis, accordingly, and representing the orientation of the sensor in the simulated MORSE environment.
  - **Video camera**[3]: This sensor generates a sequence of encoded binary character arrays to represent a set of RGBA images. The video camera sensor exports the following data fields:
    - ⋆ **Timestamp**:Returns the time passed in simulation in seconds and provides the time of capturing any particular image.
    - ⋆ **Image**: Returns the data captured by the camera as Red Green Blue Apha (RGBA) images, the size of images can be calculated in bytes as (cam_width * cam_height*4).

[2]https://www.openrobots.org/morse/doc/latest/user/sensors/pose.html
[3]https://www.openrobots.org/morse/doc/latest/user/sensors/video_camera.html

  - **Semantic camera**[4]: The semantic camera is a smart sensor which enables the process of retrieving objects instead of images. The detailed information about this camera is provided in section 3.2.2.

- **Actuators**

  - **Keyboard Actuator**[5]: The keyboard actuator can be used to control the robot movement from the keyboard. It is a simple actuator that allows moving the robot from keyboards arrows. The left and right arrows are representing left, and right turns, and up and down arrows are representing forward and backward movements accordingly. The keyboard actuator can be used to navigate the robot through its environment; however, it does not implement a realistic model of the movement.

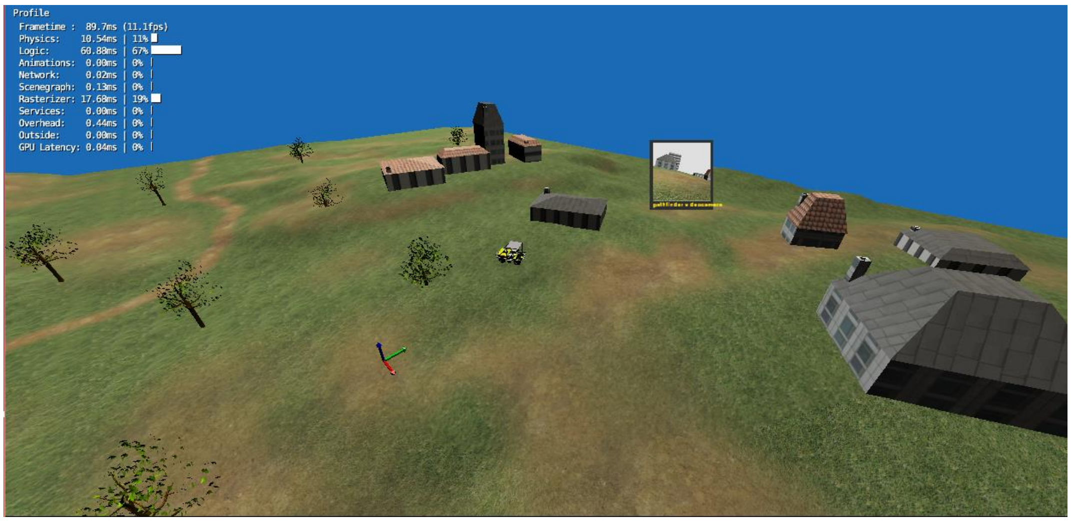
  - **Joystick Actuator**[6]: The joystick actuator can be used to control the robot movement from a joystick. Compared to the keyboard actuator this actuator provides a much greater level of realistic movement to the robot.

MORSE rendering is based on the Blender Game Engine[7], which is a 3D production open source suite, that provides advanced lighting and multi-texturing option that makes it the perfect option to make real-time interactive content. MORSE enables the user to steer a simulated mobile robot within simulated environments. Just like robotic models, MORSE provides a set of predefined outdoor and indoor 3D environment models that can be used for different scenarios and applications. Moreover, it enables adding new environment models. After equipping AMOR with all its necessary sensors and actuators, it has been placed in an initial position of a predefined 3D outdoor environment model of MORSE. The chosen environment is called trees.blend, it contains a set of landmark objects, namely a set of trees and buildings Fig.3.3. shows AMOR in its environment.

3.2.2 Semantic camera and object annotation

Semantic segmentation and labeling, place classification, and object detection and recognition are prerequisites for creating semantic information from sensor inputs. The advancement in these areas as described in section 2.3 in the works of (Stückler et al., 2015; Filliat et al., 2012; Chen and Asawa, 2017; Girshick et al., 2014; Chen et al., 2014; Long et al., 2015) are indications of their importance. With rapid advancement in these areas, one can expect the development of a semantic camera in the near

[4]https://www.openrobots.org/morse/doc/latest/user/sensors/semantic_camera.html
[5]https://www.openrobots.org/morse/doc/latest/user/actuators/keyboard.html
[6]https://www.openrobots.org/morse/doc/latest/user/actuators/joystick.html
[7]https://www.blender.org/features/game-creation/



```
Profile
  Frametime :  89.7ms (11.1fps)
  Physics:    10.54ms | 11%
  Logic:      60.88ms | 67%
  Animations:  0.00ms | 0%
  Network:     0.02ms | 0%
  Scenegraph:  0.13ms | 0%
  Rasterizer: 17.68ms | 19%
  Services:    0.00ms | 0%
  Overhead:    0.44ms | 0%
  Outside:     0.00ms | 0%
  GPU Latency: 0.04ms | 0%
```

Fig. 3.3: AMOR in an initial position of its simulated environment.

future, which will be a camera that detects objects and provides detailed information about them. This research marks the beginning of this new technology, both its path of development and possibilities for the future.

MORSE provides the semantic camera that emulates a very high-level camera, however, it outputs objects located within its field of view. It can be defined as a smart sensor that retrieves information instead of images (Morse Documentation, 2010). For the semantic camera to be able to detect objects in the MORSE environment, it is essential to set objects to interactive passive objects[8]. The semantic camera can grasp passive objects in the environment. To make an object passive, its property object needs to be set to "object" and its value to "True." A passive object can be enriched and annotated with an additional set of optional properties; these properties are as follows:

- **Label:** A string value that is used to show the objects name when detected by the semantic camera. If the name is not given, the original Blender name is retrieved.

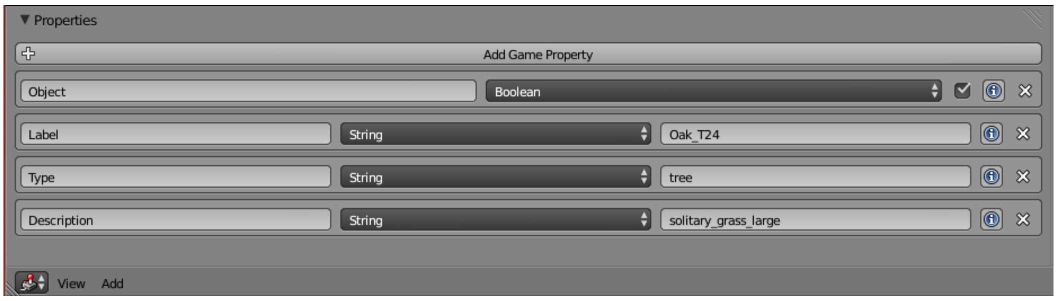
- **Description:** A string value that is used to give a more detailed description to the object.

- **Type:** A string value that is used to provide a type to the object.

- **Graspable:** A Boolean value that should be set to True in order for the object to be detected by the semantic camera.

[8]http://www.openrobots.org/morse/doc/1.4/user/others/passive_objects.html

An illustration of object annotation is shown in Fig.3.4, in which all passive object properties have been utilized to provide a complete definition of the object. The object has been provided with a name, description and type and its graspable value has been set to True which enables the semantic camera to detect it when it is present in its field of view.



Object	Boolean	✓
Label	String	Oak_T24
Type	String	tree
Description	String	solitary_grass_large

Fig. 3.4: An annotation of an object using the Blender Game Engine.

Likewise, all landmark objects have been manually annotated with a set of properties namely, name, description and type and also their graspable values have been set to True. Fig.3.5 illustrates an overview of the landmarks of the environment.

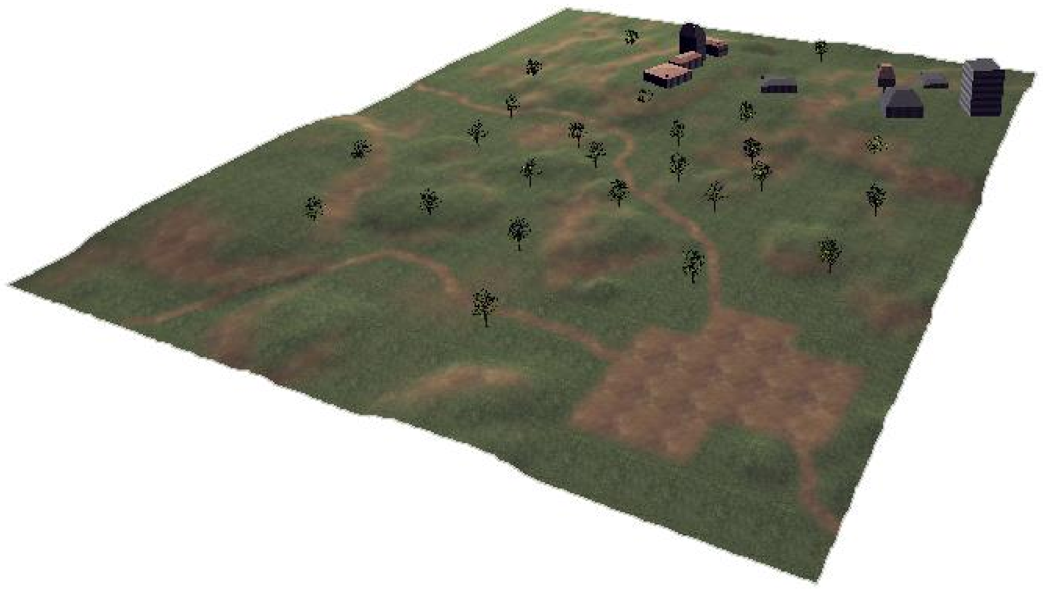


Fig. 3.5: An overview of all annotated landmark objects.

During navigation, the semantic camera is able to detect and retrieve the following information of active-passive objects placed in its field of view:

- **Name:** A string value represents the name of the object.
- **Type:** A string value represents the type of the object.
- **Description:** A string value represents the object's description.

- **Position:** A vec3, float type variable represents the exact position of the object in the environment.

- **Orientation:** A quaternion represents the orientation of the object in the environment.

3.3 Robot Operating System, and AllegroGraph

To compensate for the gap between the real and the simulated model of AMOR, the same robotic middleware of the real AMOR has been utilized for the simulated AMOR. Hence, the Robot Operating System (ROS)[9], which is used by the real model of AMOR, has been used by the simulated AMOR. Thus, allowing a very simple porting of the software developed in the simulation to the real robot.

ROS is an open source operating system that provides hardware abstraction, package management and a message passing process. Moreover, it offers tools and libraries to support the development and creation of robot applications. The obtained sensory data from sensors in the MORSE system are in the form of ROS-messages published by a so-called ROS-node. The combination of nodes create a graph; in a graph, nodes can communicate with one another using ROS-massages over ROS-topics (Tully Foote, 2018). Although ROS is language-independent, it has only three available libraries, which are Python, Lisp and C++. To be able to make use of the ROS-messages, the system is implemented as an ROS-node using ROSjava[10] messages published by MORSE system.

As described earlier, AMOR has been equipped with a set of sensors and actuators to ease its movement and data collection in its environment. However, one of the complexities of a multi-sensor system is the process of data synchronization between various sensors. For AMOR, one of the challenges is to synchronize the outputs of the video camera with its semantic camera. Since they are pointing to the same landmarks but publishing data over two different ROS-topics, it is essential to confirm that the data provided by the semantic camera and the pixel images taken by the video camera are referring to the same landmark objects. One way to ensure this is by using the timestamp and the sequence number information. The video camera provides timestamp information, showing the time that has passed in the simulation in seconds. Therefore each message published by the video camera is labeled with its timestamp. On the other hand, the semantic camera lacks this information. The following steps have been taken to overcome this problem:

[9]http://wiki.ros.org/
[10]http://wiki.ros.org/rosjava

1. Creating a custom ROS-message to be used for the sorting and filtering process.

2. Filtering duplicated messages from the semantic camera adding time-stamp, and storing the unique messages into the custom message.

3. Re-publishing custom messages over a new topic.

4. Subscribing to messages from the new topic and the video camera topic.

5. After the synchronization process, adding sequence numbers to messages and re-publishing them again.

6. Subscribing to all messages in the ROSjava.

An overview of ROS-nodes communication is illustrated and shown in Fig.3.6.

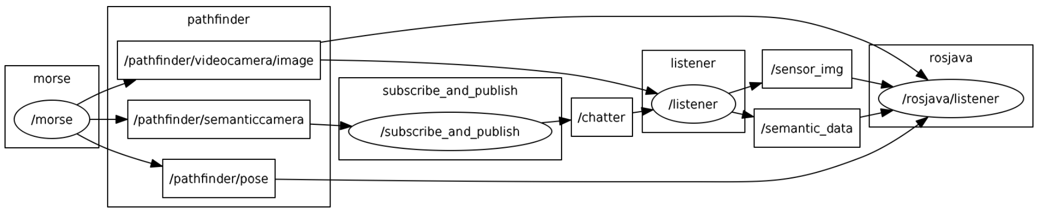


Fig. 3.6: An overview of ROS-node communication in the system.

The obtained sensory data from ROSjava is used for the creation of a set of RDF statements representing the semantic definition of the detected landmark during navigation. The resulting RDF statements have been stored in repositories in AllegroGraph[11]. AllegroGraph is a high-performance, semantic graph database; it enables the scaling of billions of RDF statements while maintaining its performance. AllegroGraph supports RDF, SPARQL, RDFSchema, OWL (Franz Inc, 2018a).

AllegroGraph enables services over the HTTP interface through web browsers. A tool called AGWebView, as can be seen in Fig.3.7, is provided by AllegroGraph for managing repositories, importing and exporting information as well as querying RDFs over a web browser (Franz Inc, 2018b). Another advantage of the AllegroGraph is the possibility of visualization of RDF statements in RDF graphs. AllegroGraph provides a sophisticated tool called Gruff[12]. Gruff is an interactive platform that enables data

[11]https://franz.com/agraph/allegrograph
[12]https://franz.com/agraph/gruff

visualization, SPARQL querying as well as graph querying over RDF triples (Franz Inc, 2018c). An example of a visualization done by Gruff is shown in Fig.3.8.

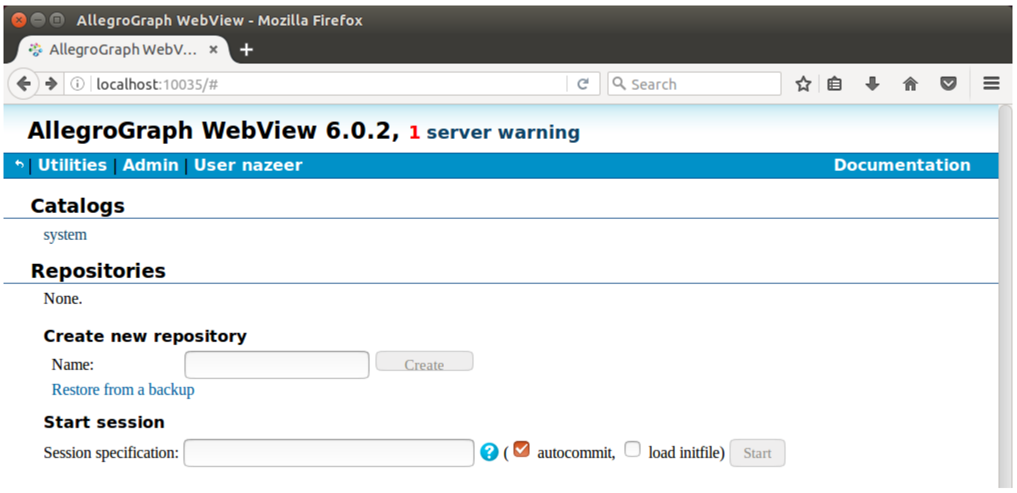


Fig. 3.7: AllegroGraph WebView server.

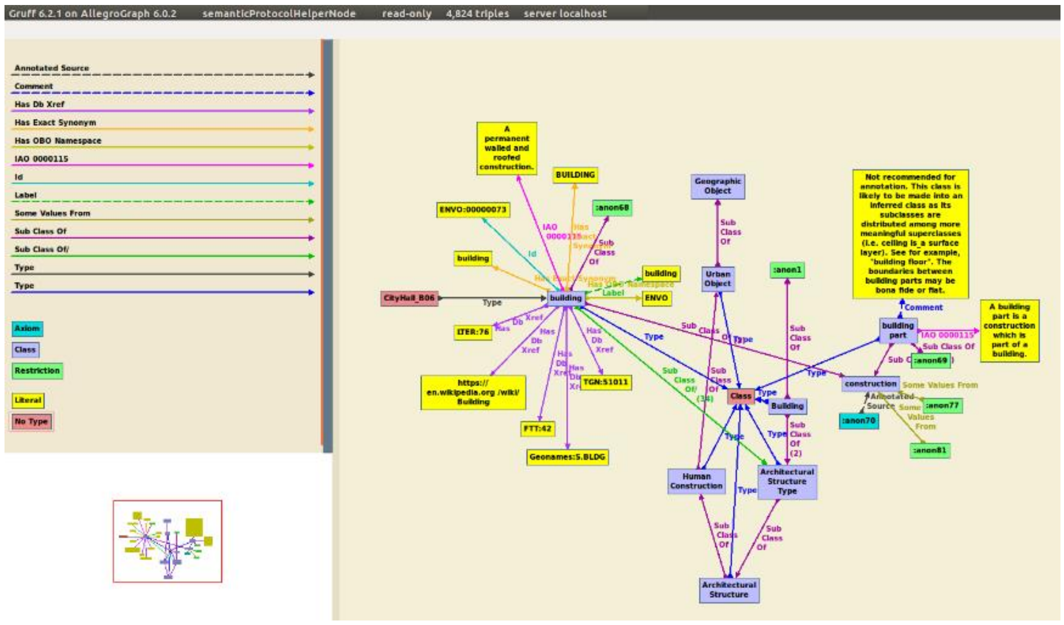

Gruff 6.2.1 on AllegroGraph 6.0.2 semanticProtocolHelperNode read-only 4,824 triples server localhost
Annotated Source
Comment
Has Db Xref
Has Exact Synonym
Has OBO Namespace
IAO 0000115
Id
Label
Some Values From
Sub Class Of
Sub Class Of/
Type
Type
Axiom
Class
Restriction
Literal
No Type
A permanent walled and roofed construction.
BUILDING
ENVO:00000073
:anon68
building
CityHall_B06
building
ENVO
ITER:76
https:// en.wikipedia.org /wiki/ Building
TGN:51011
FTT:42
Geonames:S.BLDG
Geographic Object
Urban Object
:anon1
Class
Building
Human Construction
Architectural Structure Type
Architectural Structure
Not recommended for annotation. This class is likely to be made into an inferred class as its subclasses are distributed among more meaningful superclasses (i.e. ceiling is a surface layer). See for example, 'building floor'. The boundaries between building parts may be bona fide or fiat.
building part
A building part is a construction which is part of a building.
:anon69
construction
:anon70
:anon77
:anon81

Fig. 3.8: AllegroGraph Gruff tool.

4 The Robot Semantic Protocol, RoboSemProc

Simplicity is about subtracting the obvious and adding the meaningful.

— **John Maeda**
(designer, technologist)

4.1 Introduction

The goal of this dissertation is to answer the research questions given in section 1.2. In the context of the planned research work, the task is to give a mobile robot the ability to semantically describe its environment in a human- and machine-readable form, and also to use this description for better understanding its environment. Moreover, to use the description for communication with humans in a natural language through a tailored, natural language empowered interface. To this end, a novel approach called RoboSemProc will be proposed. The RoboSemProc employs the application of semantic technology in robotics and uses ontologies as a medium for knowledge representation and communication while proposing a platform to facilitate natural language communication.

This section describes the requirements and different stages of the RoboSemProc. An overview of the requirement needed for the proposed concept is illustrated and shown in Fig.4.1.

As can be seen in Fig.4.1, the main requirements for the proposed concept are classified into three main requirements. The first requirement, highlighted in the color yellow, is the process of sensory input of a mobile robot exploring an environment. The second requirement which is highlighted in green, is the semantic knowledge-base that is to be populated with semantic information modeled from sensory inputs of the mobile robot while in navigation. Finally the section in blue shows the need for a dialog system in a natural language for communication with the mobile robot, regarding the collected information of navigation. Based on these requirements a development

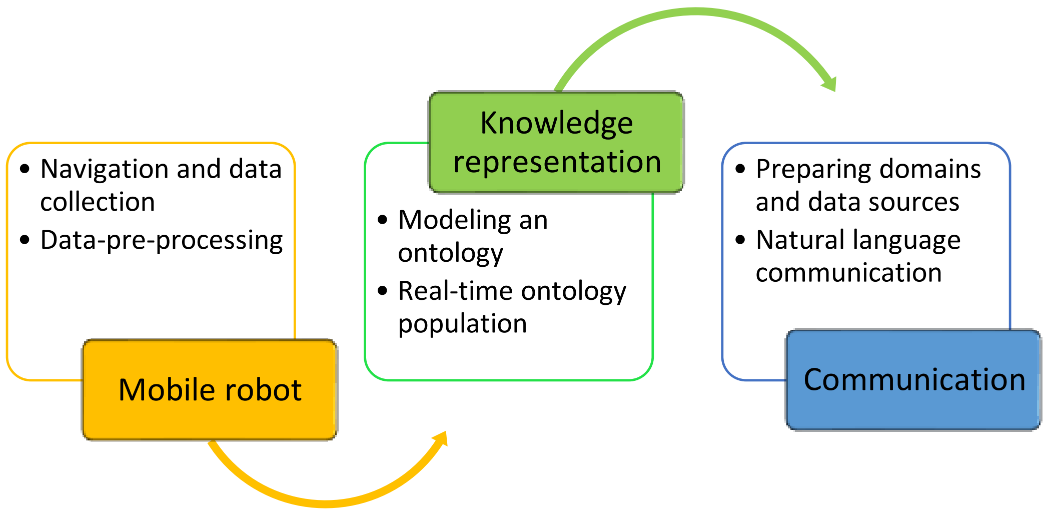

• Navigation and data collection
• Data-pre-processing
Mobile robot
Knowledge representation
• Modeling an ontology
• Real-time ontology population
• Preparing domains and data sources
• Natural language communication
Communication

Fig. 4.1: An overview of the requirements for the proposed concept.

plan has been created, as can be seen in Fig.4.2. The development plan includes the following stages:

1. Navigation and sensory data collection.
2. Sensor data pre-processing, includes filtering and synchronization of data.
3. Designing an ontology.
4. Modeling RDF relations needed for modeling RDF-statements.
5. Defining the inverse relations.
6. Navigation and data collection.
7. Real-time ontology instantiation with the semantic information representing the navigation and the environment being explored.
8. Integration of the main ontology with an external ontology to provide definition to detected landmarks of the environment.
9. Domain adaptation, creation of additional meta-data, and transforming and preparing the natural language communication system.

10. Process of updating the dialog system with information collected from the last navigation.

11. Communication with the robot.

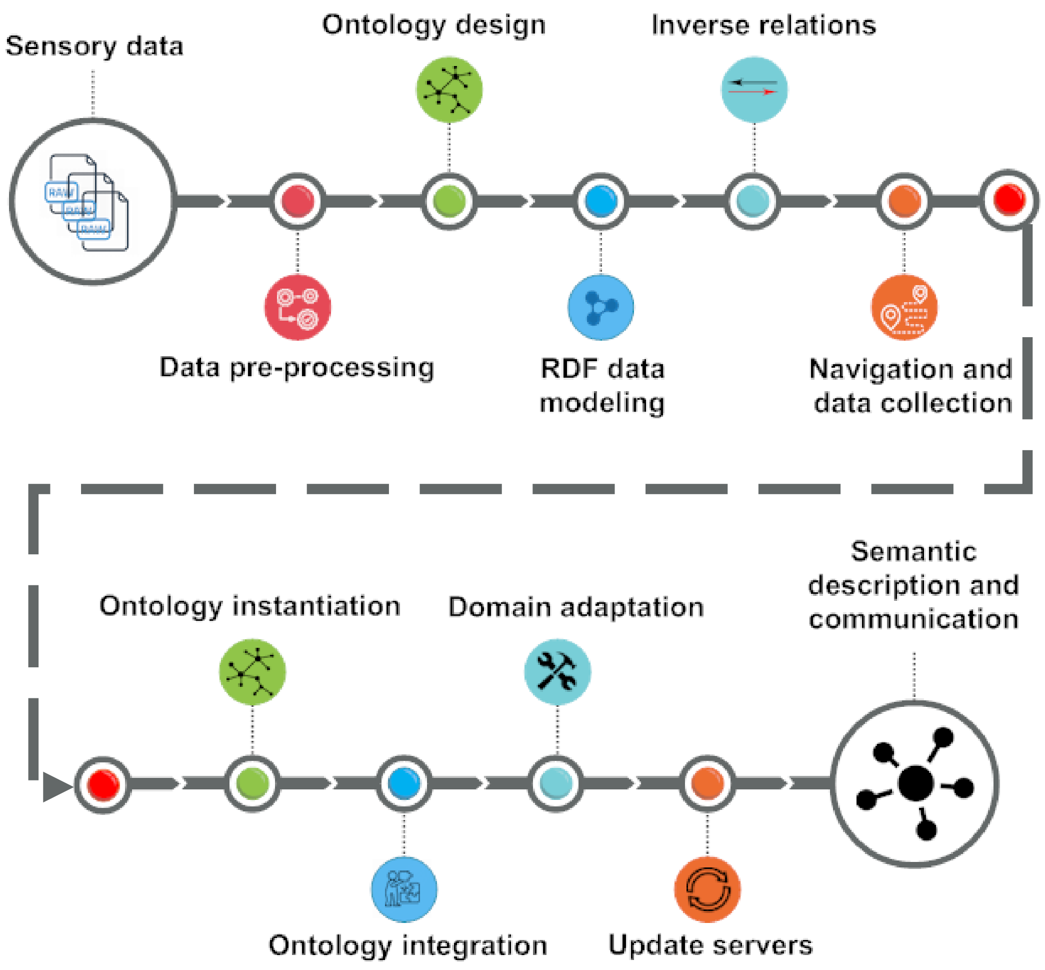


Fig. 4.2: An overview of the development stages of the proposed system in chronological order.

In the following, a description of the different stages of the development plan, as shown in Fig.4.2, is provided.

As illustrated in Fig.1.1, and described in section 2.7, different studies have been conducted to develop ontologies for knowledge representation in robotics. Examples are the work of (Fiorini et al., 2017; Kuipers et al., 2000; Paull et al., 2012; Plauska, 2013), and surveys presented in the work of (Plauska, 2013; Compton et al., 2009b) show some example of these ontologies. However, those ontologies are fixed and present robots with pre-defined information, that can be used by robots to perform a particular task, and need to be instantiated with new individuals manually whereby, each time new information is provided. Despite their advantages, due to the diversity of applications of mobile robots in dynamic and even unknown environments, modeling pre-defined ontologies and providing robots with pre-defined information is an insufficient and a tedious task. The RoboSemProc proposed in this work however, defines an ontology as a medium for semantic knowledge representation, in line with an instantiation concept which populates the ontology in real-time by sensory outputs of a mobile robot. In this way, the ontology is instantiated in real-time by the mobile robot itself. This approach enables the mobile robot to enrich the ontology with additional information on each exploration. Accordingly, and as described in section 4.2, the AMOR core ontology has been modeled to be instantiated in real-time with the AMOR tour information as well as its perception of its explored environment.

To achieve this, a set of relations and their inverse relations for modeling RDF-statements have been defined. These relations represent the tour of AMOR and its explored environment semantically. Detailed information about these relations is given in sections 4.3, 4.4, 4.5 and 4.7.

The simulated environment which has been chosen for the experiment consists of two types of landmarks, namely trees and buildings. To provide more detailed information about building type landmarks an external ontology called ENVO ontology has been integrated into the AMOR core ontology. This integration is an indication for the possibility of the future extension and enriching the AMOR core ontology proposed by this work with external ontologies. Detailed information is given in section 4.10.

As described in the state-of-the-art in section 2.8, different QA systems have been proposed and used for natural language communication. However none of those systems have been used for communication between humans and robots, they rather answer questions over pre-defined static data sources. This work, however, shows the application of QA in the field of robotics by utilizing the AMOR core ontology as a data source for natural language communication. To achieve this and to be able to use the data collected from the exploring for communication, the AMOR core ontology has been enriched with additional meta information in real-time while exploring. This additional information has been stored in different repositories of the AllegroGraph server. The AllegroGraph server is an ontology based server which has been utilized to store the AMOR core ontology as well as different additional information created in real-time by AMOR; detailed information is given in section 3.3. This information has been used by YodaQA system for natural language communication. The YodaQA system is a QA system that uses structured and unstructured data sources for natural language communication; details are given in section 4.11.

Based on the development plan an overview of RoboSemProc is illustrated and shown in Fig.4.3. In the following sections, the stages of the system are described in detail.

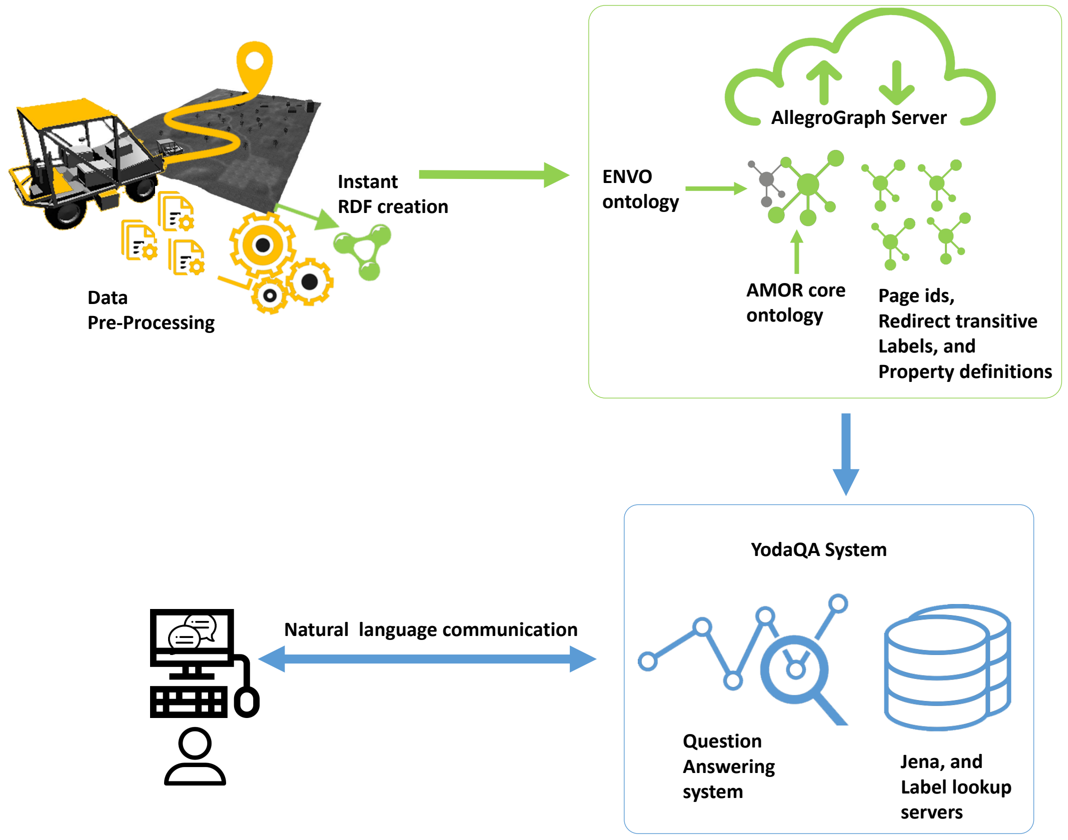


Fig. 4.3: An overview of the RoboSemProc.

4.2 AMOR Core Ontology

Due to their importance and application for formal and semantic representation of the real-world, modeling ontologies require greater attention. Ontologies can be varied in size from being simple and small or complex and huge, their size and level of complexity largely depend on their desired applications and domains. Smaller and simpler ontologies are easier to scale and are least computationally expensive, yet the complexity cannot be avoided when modeling an ontology for a larger domain and targeting broader audiences. Ontology engineering is concerned with modeling simpler and yet detailed and complete ontologies (Holsapple and Joshi, 2002; Mallak et al., 2017). Different approaches can be taken for modeling an ontology, in the work of Holsapple and Joshi (2002) five different approaches for modeling ontologies are introduced as follows,

- **Inspirational approach:** In the inspirational approach, the need for an ontology for a particular domain of interest must be addressed. The developer has to use his/her creativity, imagination, and views to emphasize the importance of an ontology for the particular domain and how it should meet the recognized need.

In this approach, the desired ontology is designed based on the personal view of the developer on the domain of interest.

- **Collaborative approach:** In a collaborative approach, an ontology is designed from the point of views of a joint group of experts. In this approach, the collection of ideas and creativity of a group results in the avoidance of any blind spots which are often presented in the inspirational approach.

- **Synthetic approach:** In this approach, an ontology is designed from a unified set of ontologies with the constraint that none of the selected ontologies are subsuming each other and are, accompanying new sets of concepts.

- **Inductive approach:** In the inductive approach, a given ontology is examined, analyzed and observed for a singular case, and the same ontology is then considered and applied to other similar cases.

- **Deductive approach:** In the deductive approach, an ontology in designed by distilling and filtering general concepts from an ontology. The resulting ontology is applied to a specific case and a distinct domain.

This work utilizes the synthetic approach for modeling ontologies, hence, the AMOR core ontology has been modeled based on the CORA ontology. As described in section 2.7 in the work of (Fiorini et al., 2017), the CORA ontology is the IEEE standard ontology introduced by the ORA group, which itself is an extension of the SUMO ontology. The AMOR core ontology extends the CORA ontology by adding new classes and hierarchies to represent the movement and vision of AMOR while navigating in its environment. The class hierarchy of the AMOR core ontology is shown in Fig.4.4.

The AMOR core ontology has been created with the Protégé[1] software. It has been designed as a well-defined, tailored ontology to represent all necessary concepts needed to semantically describe the mobile robot movement and its vision of its environment. Fig.4.5 shows the ontology in the Protégé software.

4.3 Defining pose nodes

One basic fundamental of the semantic environment description is describing AMOR's status and its movement in its environment. Consequently, AMOR-Pose nodes are considered as the primary nodes of the system. Acknowledging there is only one AMOR navigating in the environment, a single AMOR node is required to represent

[1]https://protege.stanford.edu/

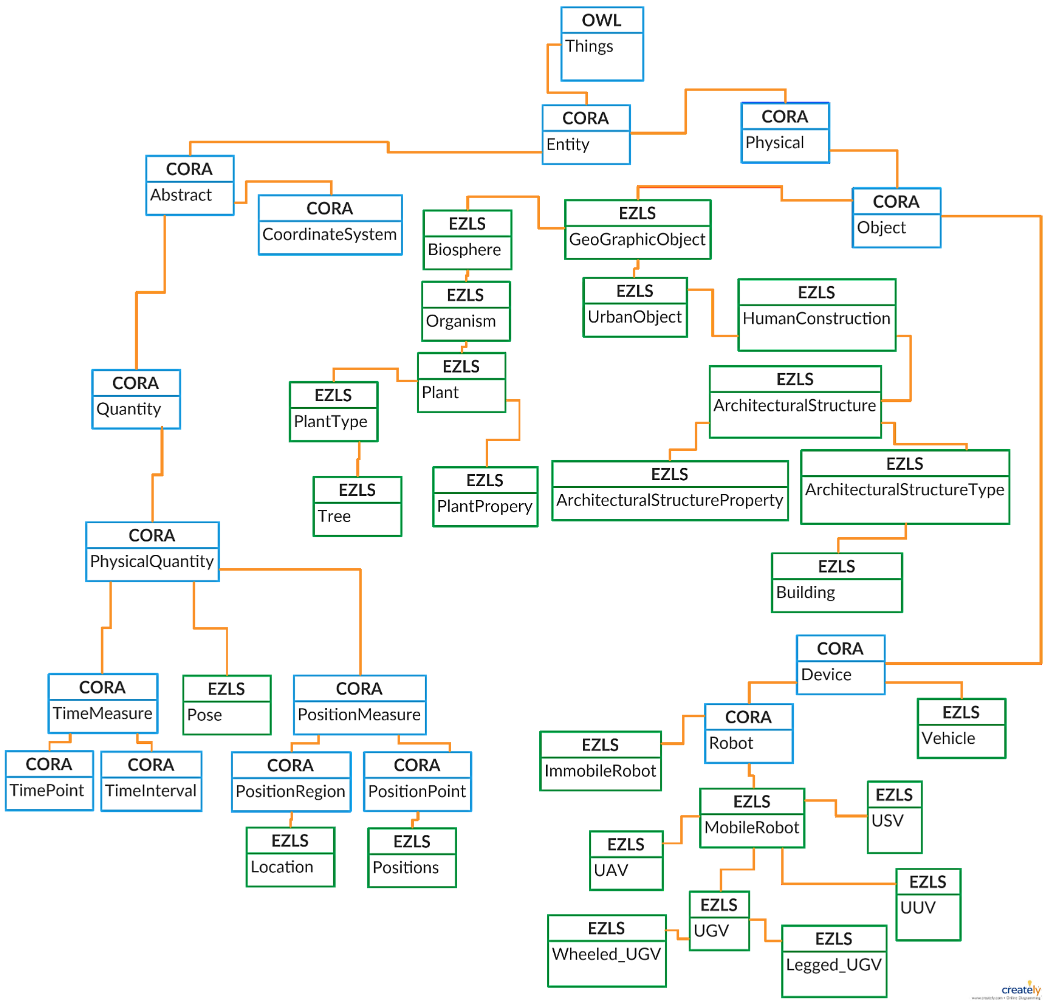
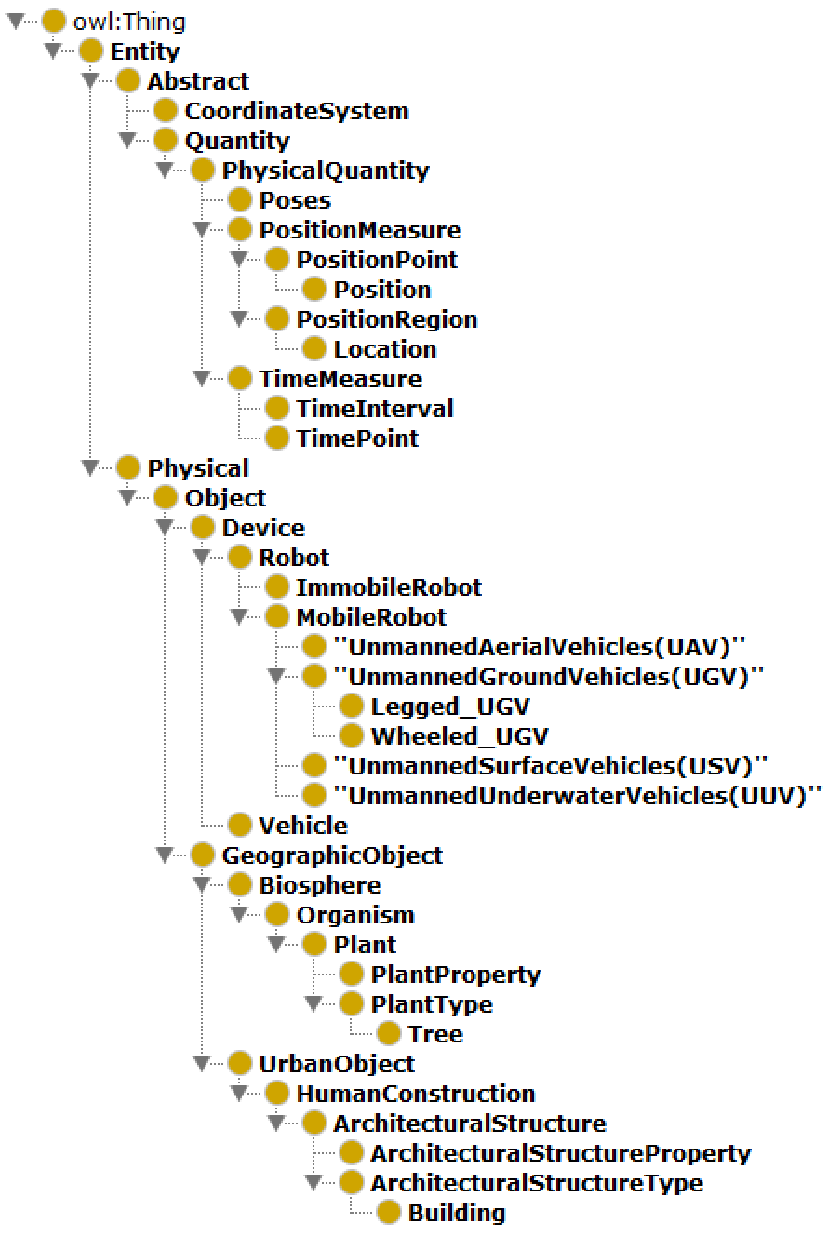


Fig. 4.4: The core classes of the AMOR core ontology, in which the core classes of the CORA ontology have been utilized and extended by new classes. The classes of the CORA ontology have been marked blue and the green classes represent the new classes added to the CORA ontology to form the AMOR core ontology.

the robot, yet, there can be an infinite number of AMOR-Pose nodes, representing snapshots of the AMOR's tour (Schwarte, 2017).

As mentioned in section 3.2.1, AMOR has been equipped with a pose sensor and placed in an initial position in a MORSE predefined outdoor environment. MORSE allows the user to steer the simulated AMOR through its simulated environments. During navigation the pose sensor retrieves a timestamp, position, and orientation of AMOR. The Table 4.1 shows an example of the retrieved data from the pose sensor of AMOR at a particular time.



```
owl:Thing
  Entity
    Abstract
      CoordinateSystem
      Quantity
        PhysicalQuantity
          Poses
          PositionMeasure
            PositionPoint
              Position
            PositionRegion
              Location
          TimeMeasure
            TimeInterval
            TimePoint
    Physical
      Object
        Device
          Robot
            ImmobileRobot
            MobileRobot
              "UnmannedAerialVehicles(UAV)"
              "UnmannedGroundVehicles(UGV)"
                Legged_UGV
                Wheeled_UGV
              "UnmannedSurfaceVehicles(USV)"
              "UnmannedUnderwaterVehicles(UUV)"
          Vehicle
        GeographicObject
          Biosphere
            Organism
              Plant
                PlantProperty
                PlantType
                  Tree
          UrbanObject
            HumanConstruction
              ArchitecturalStructure
                ArchitecturalStructureProperty
                ArchitecturalStructureType
                  Building
```

Fig. 4.5: Class hierarchy of the AMOR core ontology in the Protégé software.

header	pose	
	Position	orientation
seq: 72697	x: -0.272159099579	x: 0.00279031158425
secs: 1517232973	y: -0.746236026287	y: -0.0080242138356
nsecs: 325983524	z: 1.12528824806	z: -0.991991996765
frame_id: /map		w: 0.12601467967

Tab. 4.1: The output of the pose sensor of AMOR.

As shown in Table 4.1, the pose sensor provides accurate coordinates for both the position and orientation of AMOR. AMOR has been modeled in such a way that even when the robot seems to be idle a small movement of its engine results in a slight change of the position. Due to the high level of accuracy provided by the pose sensor, a massive number of position coordinates are being published even when the simulated AMOR seems to be idle. To avoid complexity, the system first processes the pose information and filters unnecessary coordinates. In general, the pose information is used when a new AMOR-Pose is created. The new pose is however created in only the following two situations:

1. When the sensors of AMOR detect an object or a set of objects.

2. When no object has been detected in a time interval period, i.e., 10 seconds.

Pose nodes are numbered and stored in a repository in a triple store, the pose number of the last pose node is used to determine the pose number of the new pose node. In general, pose numbers from 1 and are incremented by one each time a new pose node is created.

4.3.1 Position, location and time of each pose

Each pose node is annotated with the time, position and location information. The time information is provided by the timestamp of the pose sensor and indicates the time passed in seconds in the navigation. This information is stored as a class with the time value and used to annotate the pose node with the time of pose creation. The predicate *hasTime* has been used to create an RDF statement by connecting the time class to its corresponding pose node.

Similarly, the position information provided by the pose sensor of AMOR is utilized to annotate the pose node with its exact position in its environment. This position information is stored as a literal value in the position class. The *hasPosition* predicate is then used to create an RDF statement describing the position of a given pose.

Based on the position coordinates, the location of each pose node has been determined. To demonstrate the environment has been classified into various rectangular areas, these areas are namely city, landscape, campus and the industrial area. The *hasLocation* predicate has been used to annotate pose nodes with their corresponding location area. Fig.4.6 illustrates an example of a pose node annotation with its time, position and location information.

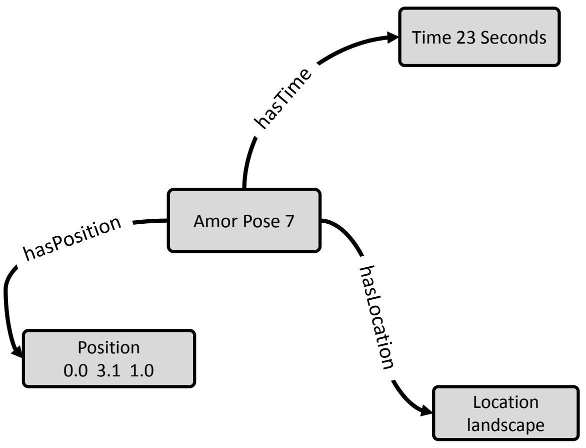


Fig. 4.6: An RDF graph describes the position, location and time of AMOR-Pose 7.

The subsequent RDF code representing the RDF example shown in Fig.4.6

Listing 4.1: RDF-statements represent the position location and time of AMORPose 7.

```
@prefix rs:<http://www.eti.uni−siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni−siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time: 23 Seconds ;
            pr:hasPosition rs:Position: 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location: landscape;
```

4.3.2 Connecting pose nodes, and the vector between them

Single navigation might result in the creation of a vast number of pose nodes, in which each pose node has been annotated with its time, position and location. Since the collection of these nodes are representing a tour or multiple tours of AMOR and describing an explored environment, it is essential to connect pose nodes to be able to make use of the collected data and gain a better understanding of the tour of the AMOR and the environment it explores.

As mentioned in section 4.3, a pose node is created either when the AMOR detects an object or a set of objects or after a period of time; this indicates that AMOR either changes its position, orientation or its time. Therefore, one can conclude that AMOR is in constant motion. This motion might happen either in place by changing its position or orientation, or in time. The *movesTo* relation, therefore, is defined as a predicate to connect a pose node to its succeeding one.

Since each pose node is provided with its exact position information, it is possible to create a predicate to illustrate the movement direction from a pose node to its subsequent one. The direction angle is calculated using the *arctan()* function, and the resulting value is stored in a literal. The *hasDirection* relation then connects the AMOR-Pose node to the literal showing the direction angle to the subsequent pose node. If there is no variation in the position of pose nodes, the literal value shows None. Fig.4.7 shows an example of the relations mentioned above.

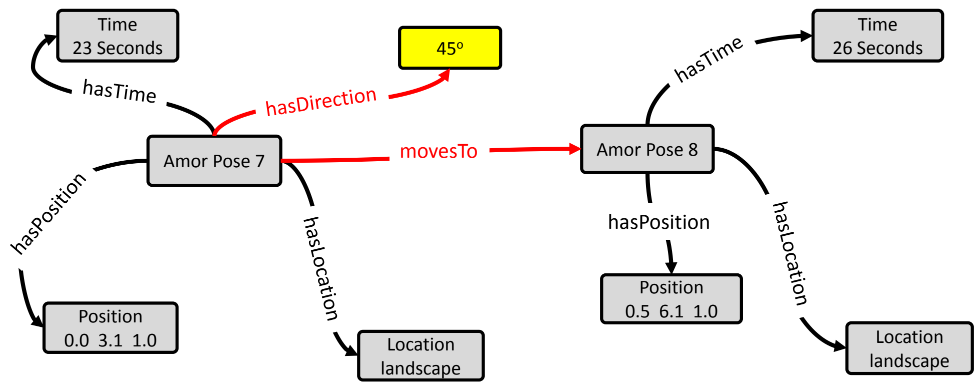


Fig. 4.7: An RDF-graph illustrating the previous relations as well as the *movesTo* and *hasDirection* relations.

The subsequent code section represents the RDF example shown in Fig.4.7

Listing 4.2: RDF-statements representing the previous relations as well as the *movesTo* and *hasDirection* relations.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time: 23 Seconds ;
            pr:hasPosition rs:Position: 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location: landscape;
            pr:hasDirection "45" ;
            pr:movesTo rs:AMORPose8;

rs:AmorPose8
```

```
            pr:hasTime  rs:Time 26 Seconds ;
            pr:hasPosition  rs:Position 0.5 6.1 1.0 ;
            pr:hasLocation  rs:location: landscape;
```

4.3.3 Moves left and right in a pathless environment

In the AMOR's given environment it is considerably challenging to determine whether AMOR drives straight forward or moves to the right or the left. These are possible to consider in a topological map with a certain road and path to follow. In a pathless environment, there is no stringently unambiguous method to determine a left-turn or right-turn (Schwarte, 2017). A good example is given in the work of Schwarte (2017) as follows,

> "A car is driving around a steep curve and then going out of it, even while the car is turning left, the driver is, however, slightly steering wheels to the right. Either information could be used to determine left or right. Moreover, it is uncertain to which degree and distance the curve and the change in the direction shall be to be considered left turn or right turn." (Schwarte, 2017, p. 31).

To overcome this problem, the derivative yaw angle has been used. Considering AMOR is a wheeled vehicle, and as shown in Fig.4.8 the following formula has been used to calculate the yaw angle.

$$\psi = \tan^{-1}\left(\frac{2(zw + xy)}{x^2 - y^2 - z^2 + w^2}\right)$$

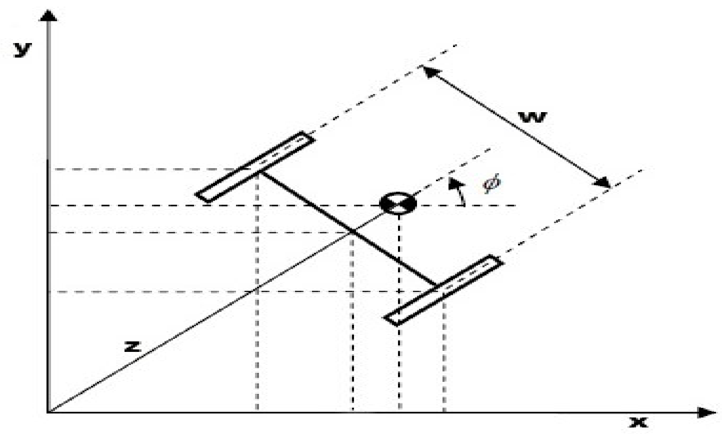


Fig. 4.8: The coordinate system and motion equations of wheeled inverted pendulum (*Yaw angle calculation* 2015).

Where x, y, and z represent the orientation axis and w represents the distance between the centers of the wheels. The formula uses the orientation of the AMOR-Pose node at the beginning of the snapshot to determine whether AMOR turns to the left or the right. The result of the yaw angle formula is stored in a literal and used to determine whether AMOR moves to the left, right or straightforward at a given pose. As shown in Fig.4.9, the *hasYawAngleDerivative* predicate has been defined to connect AMOR-Poses to a literal that contains the yaw angle information. Depending on this information, the yaw angle can be determined as steady, increasing or decreasing. A steady yaw angle is considered as moving straightforward, increasing as turning left and decreasing as turning right.

The subsequent code section represents the RDF example shown in Fig.4.9.

Listing 4.3: RDF-statements representing the previous relations as well as the movement from one pose node to another.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time 23 Seconds ;
            pr:hasPosition rs:Position 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location landscape;
            pr:hasDirection "45" ;
            pr:movesTo rs:AMORPose8;
            pr:hasYawAngleDerivative "increasingYawAngle";

rs:AmorPose8
            pr:hasTime rs:Time 26 Seconds ;
            pr:hasPosition rs:Position: 0.5 6.1 1.0 ;
            pr:hasLocation rs:Location: landscape;
```

4.4 The vision from the semantic camera

The simulated model of AMOR has been equipped with the semantic camera. As described in section 3.2.1, the semantic camera, outputs the name, type, description, position, and orientation of passive objects. In the annotation phase, all landmark objects have been set as passive objects and are all annotated with the necessary information. This information is retrieved once a landmark is placed at the field of view of the semantic camera. The retrieved data from the semantic camera describes which object has been in the vision of AMOR at a specific time, place or orientation. In other words, the semantic camera provides a semantic vision to the AMOR and plays a significant role in enabling AMOR to have a better understanding of its environment. The structure of outputs from the semantic camera is shown below.

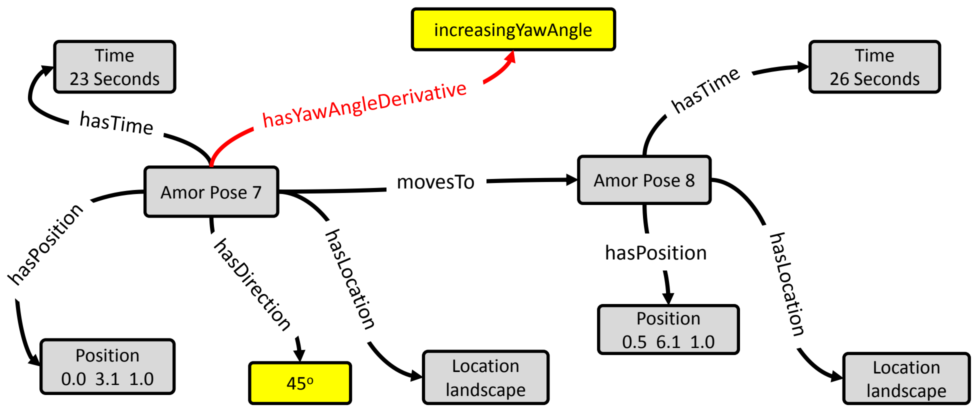


Fig. 4.9: An RDF-graph illustrating the previous relations as well as the *hasYawAngleDerivative* relation.

{"***type***": "*tree*", "***orientation***": "*y*": *0.0*, "*x*": *0.0*, "*z*": *0.1586325466632843*, "*w*": *0.9873377084732056*, "***position***": *[0.8076858520507812, -36.107215881347656, 1.061995506286621]*, "***name***": "*BigPineTree_T16*", "***description***": "*near_path_grass_large*"}.

As highlighted in the above example the semantic camera has detected an object called *BigPineTree_T16*, which is of the type *tree* and description *near_path_grass_large*", moreover the exact position and orientation of the object are also provided.

4.4.1 Landmark detection and annotation

As AMOR navigates through its environment is uses its semantic vision to detect objects of its surrounding; hence several RDF predicates have been defined to represent this vision. The first step toward the representation of the semantic vision is to assign detected objects to their corresponding pose nodes. The *sees* predicate has been used to create a set of RDF statements, representing objects perceived by AMOR at a time. The *sees* predicate as shown in Fig.4.10, is originating from a pose node pointing to a landmark detected by the pose. It is worth mentioning that a pose node may detect more than one landmark at a given time, and also a number of pose nodes can detect a specific landmark; this indicates that the landmark is visible from various positions and orientations, and also that it has been detected at different times.

As shown in Fig.4.10, AMOR at pose 7 detects two landmarks, namely Factory_B05 and LimeTree_T07. Moving to the next pose, AMOR is able to detect a new landmark called BigMaple_T03. Moreover, at this pose the Factory_B05 is still visible to AMOR and has been detected by its semantic camera. This example illustrates the possibility

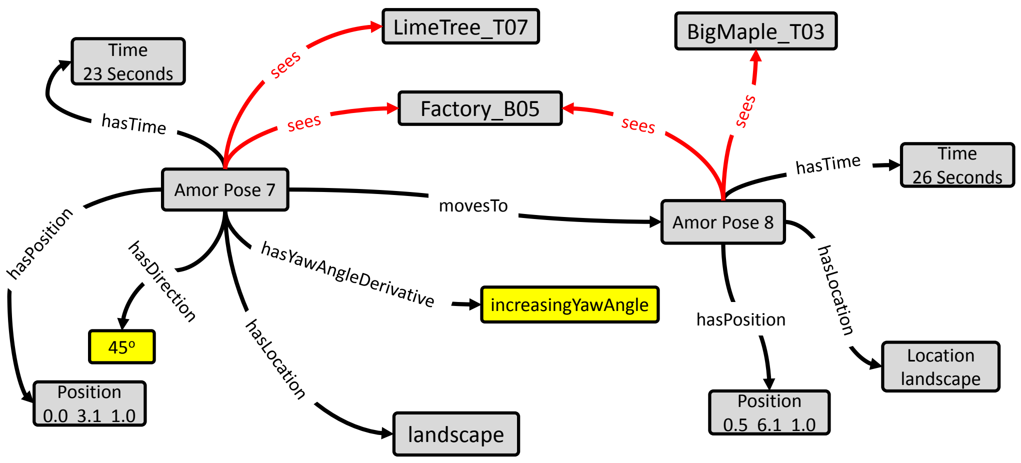


Fig. 4.10: An RDF-graph example illustrating the previous relations as well as the *sees* relation.

of detecting multiple landmarks by a pose node and also the possibility for the landmarks to be detected by multiple pose nodes.

The subsequent code section represents the RDF example shown in Fig.4.10.

Listing 4.4: RDF-statements representing the previous relations as well as the *sees* relation.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time: 23 Seconds ;
            pr:hasPosition rs:Position 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location landscape;
            pr:hasDirection "45" ;
            pr:movesTo rs:AMORPose8;
            pr:hasYawAngleDerivative rs:"increasingYawAngle";
            pr:sees rs:Factory_B05;
            pr:sees rs:LimeTree_T07;

rs:AmorPose8
            pr:hasTime rs:Time 26 Seconds ;
            pr:hasPosition rs:Position 0.5 6.1 1.0 ;
            pr:hasLocation rs:Location landscape;
            pr:sees rs:Factory_B05;
            pr:sees rs:BigMaple_T03;
```

In the annotation phase, each landmark is annotated with its type either a tree or a building and also a short description is given. The *hasType* and *hasDescription* relations are defined to enrich detected landmarks with additional meta-data information representing their type and description. The description information provides a better definition to landmarks, and the type information eases the process of querying and filtering landmarks by their type. As shown in Fig.4.11, these relations are originating from landmarks and printing to literals containing their corresponding type and description.

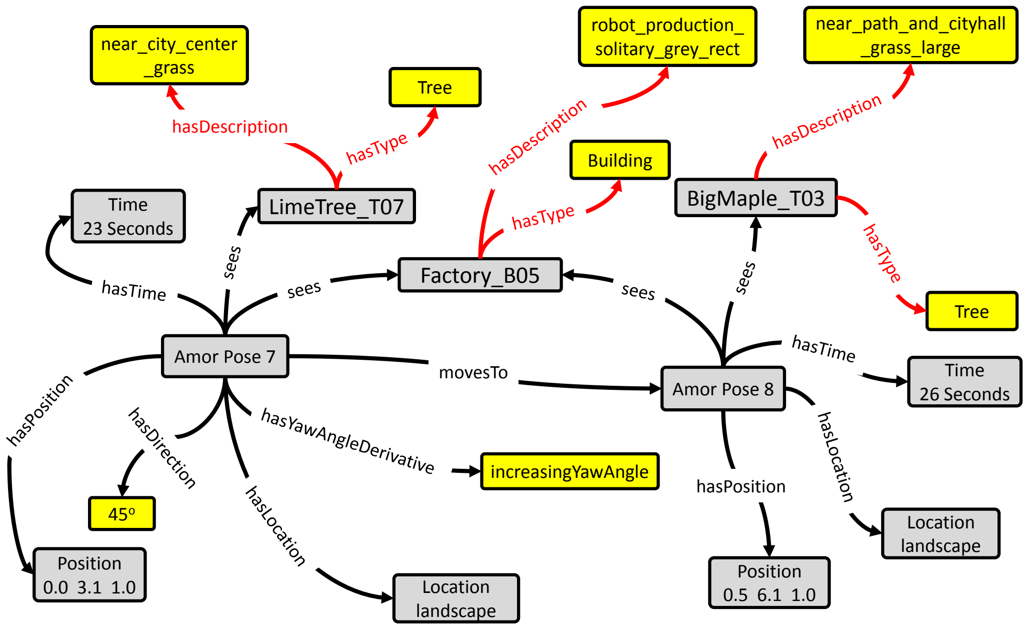


Fig. 4.11: An RDF-graph example illustrating the previous relations as well as landmark type and description.

The simulated environment contains a set of trees and buildings. Consequently, all landmarks are annotated with a type either a tree or a building. This information has been used for the naming convention and unique landmark identification. Hence, each landmark has been annotated with a unique name that contains a code which implicitly indicates its type. To illustrate and as shown in Fig.4.11, the Factory_B05 is a landmark of type building, and therefore its name contains a code starting with the letter *B*, followed by an incremental sequence number. In the same way and as shown in Fig.4.11, LimeTree_T07 and BigMaple_T03 contain a code *T07* and *T03* that indicate that they are tree number 7 and tree number 3, respectively.

The subsequent code section represents the RDF example shown in Fig.4.11.

Listing 4.5: RDF-statements representing the previous relations as well as landmark type and description.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
         pr:hasTime rs:Time: 23 Seconds ;
         pr:hasPosition rs:Position: 0.0 3.1 1.0 ;
         pr:hasLocation rs:Location landscape;
```

```
              pr:hasDirection "45" ;
              pr:movesTo rs:AMORPose8;
              pr:hasYawAngleDerivative rs:"increasingYawAngle";
              pr:sees rs:Factory_B05;
              pr:sees rs:LimeTree_T07;

rs:AmorPose8
              pr:hasTime rs:Time 26 Seconds ;
              pr:hasPosition rs:Position: 0.5 6.1 1.0 ;
              pr:hasLocation rs:Location landscape ;
              pr:sees rs:Factory_B05 ;
              pr:sees rs:BigMaple_T03 ;

rs:Factory_B05
              pr:hasType "Building" ;
              pr:hasDescription "robot_production_solitary_grey_rect" ;

rs:LimeTree_T07
              pr:hasType "Tree" ;
              pr:hasDescription "near_city_center_grass" ;

rs:BigMaple_T03
              pr:hasType "Tree" ;
              pr:hasDescription "near_path_and_cityhall_grass_large" ;
```

4.4.2 Landmarks' position and location

As described in previous sections, the manual annotated information has been used to create a number of relations such as the *sees*, *hasType*, and *hasDescription* relations. In this section, however, the position information of the landmarks is utilized to create a number of RDF relations. This is just like the pose sensor of AMOR, which outputs the position and orientation of AMOR and has been used to create a number of relations describing the position, location, direction, and yaw angle derivative of AMOR-Pose nodes. The semantic camera also outputs the position information of the detected landmarks and can be used to create a number of relations describing the position, location and the vicinity type of landmarks as well as their inter-object relationship. It is worth mentioning that the position information is not manually annotated to landmarks; it is provided automatically by the semantic camera. The position information is a literal value representing the exact coordinate position of the landmark in the environment. The *hasPosition* relation, as shown in Fig.4.12, is originating from landmarks and pointing to its corresponding position information to create an RDF-statement that describes the exact position of the landmark in the simulated environment.

As mentioned earlier, in the annotation phase, the simulated environment has been divided into four different rectangle areas, leading to the creation of four distinct locations. Depending on the position coordinate information, the location of the landmark in the environment is determined. The *hasLocation* predicate, as shown in Fig.4.12, is then used to connect the landmark to its corresponding location.

The subsequent code section represents the RDF example shown in Fig.4.12.

Listing 4.6: RDF-statements representing the previous relations as well as the position and location of landmarks.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time: 23 Seconds ;
            pr:hasPosition rs:Position: 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location: landscape;
            pr:hasDirection "45" ;
            pr:movesTo rs:AMORPose8;
            pr:hasYawAngleDerivative rs:"increasingYawAngle";
            pr:sees rs:Factory_B05;
            pr:sees rs:LimeTree_T07;

rs:AmorPose8
            pr:hasTime rs:Time: 26 Seconds ;
            pr:hasPosition rs:Position: 0.5 6.1 1.0 ;
            pr:hasLocation rs:Location:landscape ;
            pr:sees rs:Factory_B05 ;
            pr:sees rs:BigMaple_T03 ;

rs:Factory_B05
            pr:hasType "Building" ;
            pr:hasDescription "robot_production_solitary_grey_rect" ;
            pr:hasPosition "0.6 5.5 1.0" ;
            pr:hasLocation rs:landscape;

rs:LimeTree_T07
            pr:hasType "Tree" ;
            pr:hasDescription "near_city_center_grass" ;
            pr:hasPosition rs:Position: 1.0 4.2 1.0 ;
            pr:hasLocation rs:Location: landscape;

rs:BigMaple_T03
            pr:hasType "Tree" ;
            pr:hasDescription "near_path_and_cityhall_grass_large" ;
            pr:hasPosition rs:Position: 1.7 6.1 1.0 ;
            pr:hasLocation rs:Location: landscape;
```

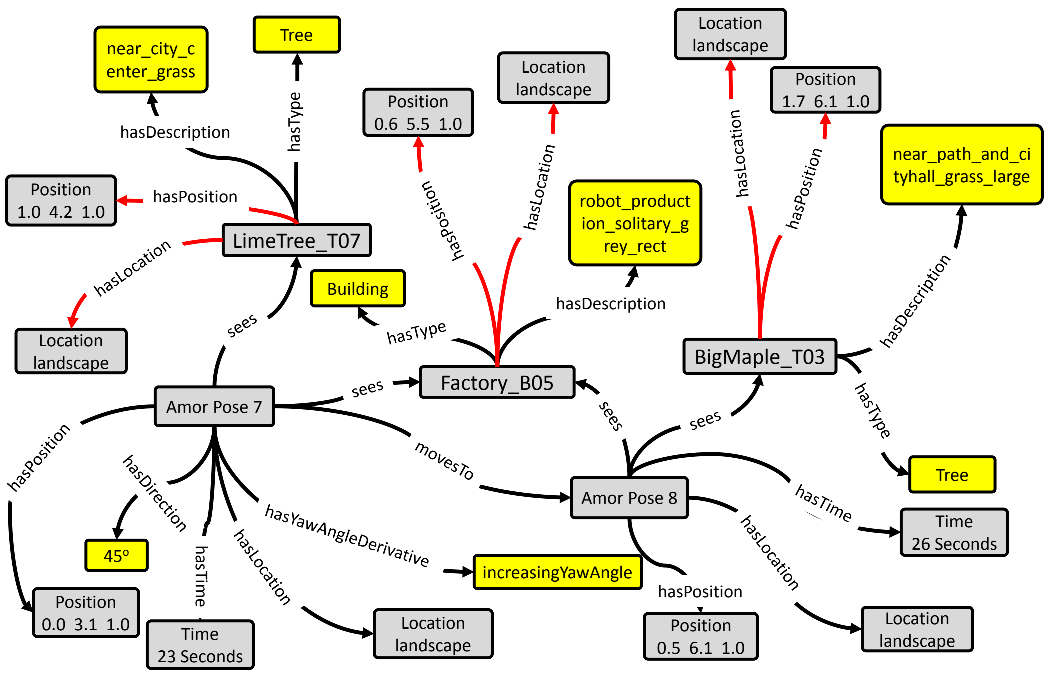


Fig. 4.12: An RDF-graph example illustrating the previous relations as well as the position and location of landmarks.

4.4.3 Landmarks and their type of vicinity

AMOR is provided with the position information of all detected landmarks; this information is used to calculate the Euclidean distance between a landmark to its neighbors. Considering the first landmark *Fl* and the second landmark *Sl*, the following formula has been used to determine the vicinity value between them.

$$V(Fl, Sl) = \sqrt{(Fl_x, Sl_x)^2 + (Fl_y, Sl_y)^2 + (Fl_z, Sl_z)}$$

Based on the vicinity value (the Euclidean distance) the vicinity type of landmarks is determined; hence landmarks are categorized into the following categories, close proximity, medium proximity and solitude. Since a new landmark can be detected during navigation, the vicinity value is constantly calculated and updated each time a pose node detects a new landmark. Therefore, the vicinity type of the landmark depending on its new vicinity value can be updated from solitude to medium proximity or eventually to close proximity.

The vicinity value is stored as a literal and the *withVicinityType* is defined to create RDF-statements describing the vicinity type of landmarks. As shown in Fig.4.13, the *withVicinityType* predicate is originating from a landmark and pointing to a literal that contains its corresponding vicinity type.

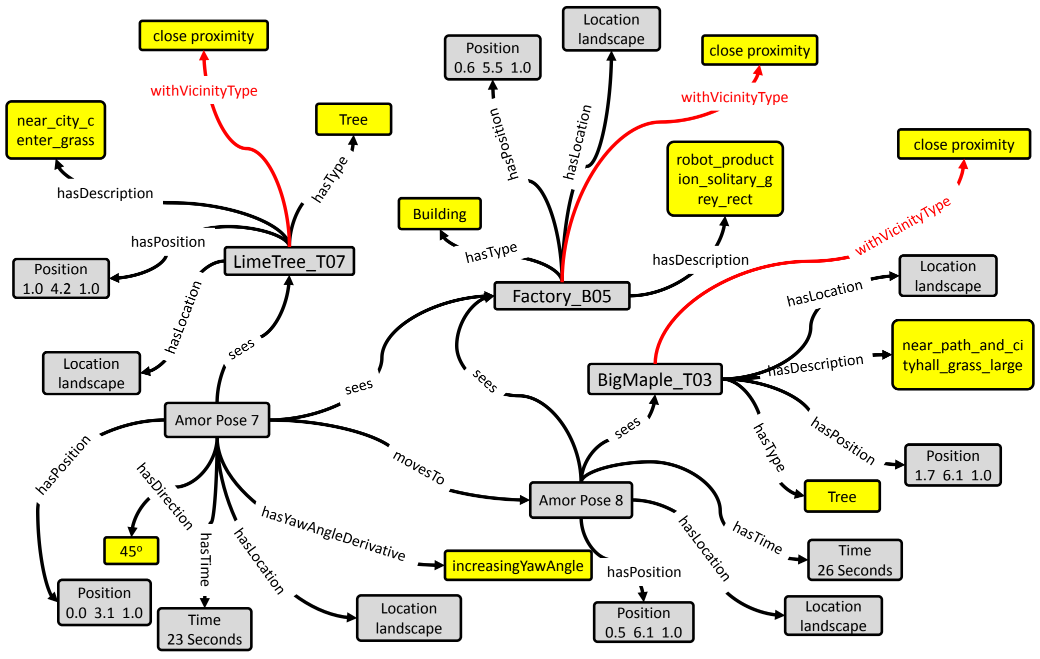


Fig. 4.13: An RDF-graph example illustrating the previous relations as well as the type of vicinity of landmarks.

The subsequent code section represents the RDF example shown in Fig.4.13.

Listing 4.7: RDF-statements representing the previous relations as well as type of vicinity of landmarks.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AmorPose7
            pr:hasTime rs:Time: 23 Seconds ;
            pr:hasPosition rs:Position: 0.0 3.1 1.0 ;
            pr:hasLocation rs:Location:landscape;
            pr:hasDirection "45" ;
            pr:movesTo rs:AMORPose8;
            pr:hasYawAngleDerivative rs:"increasingYawAngle";
            pr:sees rs:Factory_B05;
            pr:sees rs:LimeTree_T07;

rs:AmorPose8
            pr:hasTime rs:Time: 26 Seconds ;
            pr:hasPosition rs:position: 0.5 6.1 1.0 ;
            pr:hasLocation rs:Location: landscape ;
            pr:sees rs:Factory_B05 ;
            pr:sees rs:BigMaple_T03 ;

rs:Factory_B05
            pr:hasType "Building" ;
            pr:hasDescription "robot_production_solitary_grey_rect" ;
            pr:hasPosition rs:Position: 0.6 5.5 1.0" ;
            pr:hasLocation rs:Location: landscape;
            pr:withVicinityType "close proximity";

rs:LimeTree_T07
            pr:hasType "Tree" ;
            pr:hasDescription "near_city_center_grass" ;
            pr:hasPosition rs:Position: 1.0 4.2 1.0 ;
            pr:hasLocation rs:Location: landscape;
            pr:withVicinityType "close proximity";

rs:BigMaple_T03
            pr:hasType "Tree" ;
            pr:hasDescription "near_path_and_cityhall_grass_large" ;
            pr:hasPosition rs:Position: 1.7 6.1 1.0" ;
            pr:hasLocation rs:Location: landscape;
            pr:withVicinityType "close proximity";
```

4.5 Environmental perception and modeling complex inter-object relations

In RDF, a binary relation or what is considered a statement or a triple is created by linking two individuals (resources) or an individual and a value (literal). All mentioned RDF statements have been created by implementing the binary relation concept, in which either two resources or a resource and a literal have been linked with a single predicate to model a single RDF-statement. However, in some cases, more than two individuals are required to model a single relation, this becomes more apparent in a dynamic environment when RDF-statements are created from multiple data sources. Modeling inter-object relations is a good example, in which a number of sensors and objects are involved in modeling a single RDF-statement.

Over the last decades, different approaches have been proposed to model RDF n-ary relations; the detailed information is given in section 2.5.1.1. Among the described approaches, the blank-node and reification relations are the most popular. However, they both encounter a number of drawbacks and disadvantages that make them unreliable approaches for modeling complex RDF statements. As described in section 2.5.1.1, a considerable number of studies have been conducted to address these drawbacks, and a number of approaches have been proposed too, yet none of them have been considered as alternative approaches. Consequently, the problem of modeling complex n-ary relation remains an open problem.

In this work, a new approach called the helperNode approach has been introduced for modeling complex n-ary relations. In the helperNode approach, a single node is used to represent all relations between two related nodes. As shown in Fig.4.12, each landmark is provided with a code that is used to determine the landmarks type implicitly, and it is also used to identify the landmark uniquely. In the helperNode approach, a new node is created from the combination of codes of two related nodes. For instance, the helperNode created from the following nodes LimeTree_T07 and BigMaple_T03 is called T07T03. It is worth mentioning that the sequence in which these codes are combined is essential, for instance, T07T03 and T03T07 are both helperNodes created to model relations between LimeTree_T07 and BigMaple_T03, but they are not identical. Therefore, in the helperNode approach, the code of the landmark to be located which is called primary landmark is placed first followed by the code of the landmark, that is in relation to the primary landmark which is called the reference landmark. The primary landmark is connected to the helper node with the *towardsHelperNode* predicate, and a predicate called *fromHelperNode* connects the helperNode to the reference landmark.

In the following sections the application of the blank-node, reification, and helperNode approaches for modeling complex relations is given. Experimental navigation is then conducted to examine each of these approaches and in section 4.6 an evaluation of these approaches is presented.

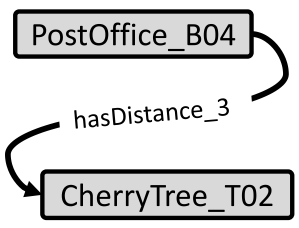
4.5.1 The distance relation between two landmarks

The position information of landmarks is used for computing the distance between them. While navigating whenever AMOR detects more than a landmark at any given pose node, the distance between them is computed and stored in a literal. The distance information is then used to model RDF statements that describe the distance between two landmarks. Modeling the distance relation requires more than two nodes; hence, this relation is considered a complex relation. The following three approaches illustrate different approaches for modeling the distance relation between two landmarks.

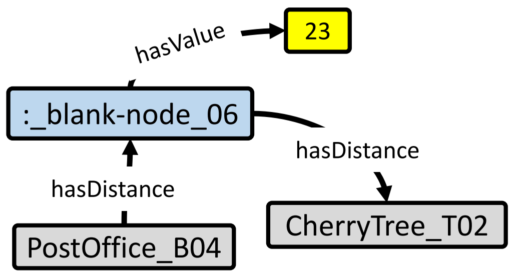
- **Binary approach:** in a binary approach, a relation is created by linking two nodes, either two individuals or an individual and a value. The distance relation, however, includes more than two nodes, it contains two individuals and a value. Therefore, to be able to use the binary approach for modeling the distance relation, the number of nodes involved in the relation must be reduced to only two nodes. As mentioned earlier the distance is a literal value. Therefore, as shown in Fig.4.14a, it is possible to eliminate this node from the relation and represent its value along with the predicate. Although the binary approach can be applied to model distance relation in this way, it is not recommended to store values or extra information as part of the predicate. This is due to the fact that adding extra information to the predicate will lead to the creation of a huge number of predicates which results in increasing the size and complexity of the overall ontology.

- **Blank-node approach:** In the blank-node approach, an anonymous blank-node is added to the relation. As shown in Fig.4.14b, a *hasDistance* predicate is created and originates from the primary landmark pointing to the blank-node. The blank-node then refers to the relation to the literal showing the distance value with the *hasValue* predicate and the reference landmark with the *hasDistance* predicate.

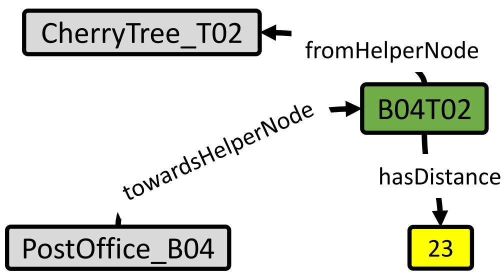
- **The helperNode approach:** In this approach, as shown in Fig.4.14c, an additional node called the helperNode is created from the codes of both the primary and reference landmarks. The *towardsHelperNode* predicate is created and originated from the primary landmark pointing to the helperNode, and the *fromHelperNode* predicate is created and originated from the helperNode to the reference landmark. This way both related landmarks are connected sequentially. The helperNode then references the relation to the literal that contains the distance value with the *hasDistance* predicate.



(a) Binary approach



(b) Blank-node approach



(c) helperNode approach

Fig. 4.14: An illustration of the distance relation between the cherry tree and the post office in different approaches.

The subsequent code section represents the binary approach for the modeling distance relation between to landmarks as shown in Fig.4.14a.

Listing 4.8: RDF-statements representing the binary approach for modeling distance relations between the cherry tree and the post office.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:hasDistance_3 rs:CherryTree_T02 ;
```

The subsequent code section represents the blank-node approach for modeling the distance relation between two landmarks as shown in Fig.4.14b.

Listing 4.9: RDF-statements represent the blank-node approach for modeling the distance relation between the cherry tree and the post office.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:hasDistance  _:blank-node_06 ;
_:blank-node_06
            pr:hasDistance  rs:CherryTree_T02 ;
            pr:hasValue  "3" ;
```

The subsequent code section represents the helperNode approach for modeling the distance relation between two landmarks as shown in Fig.4.14c.

Listing 4.10: RDF-statements representing the helperNode approach for modeling distance relation between the cherry tree and the post office.

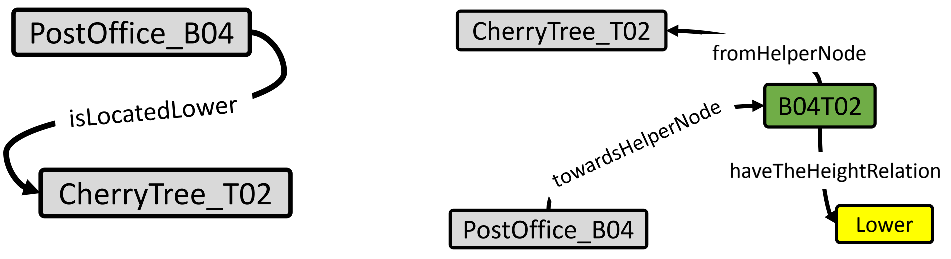
```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:towardsHelperNode  rs:B04T02 ;
rs:B04T02
            pr:fromHelperNode  rs:CherryTree_T02 ;
            pr:hasDistance  "3" ;
```

4.5.2 The height relation between two landmarks

To determine the height difference between two landmarks the z coordinate information has been utilized. Just like distance, the height difference between two landmarks has been computed whenever AMOR has detected more than a landmark in a given pose node. Unlike the distance relation, the height relation involves only two individuals, and therefore it is not considered a complex relation. This relation can be modeled merely using the basic binary approach without the need of any modification. However, as mentioned in section 4.5.1, the helperNode is created to represent all relations between two related landmarks and to be able to describe all possible relations between related landmarks, the helperNode approach has also modeled the height relation.

According to the z coordinate information, three different height relations have been defined; the primary landmarks can be located higher, lower or at the same level as the reference landmark. Consequently, in a binary approach three predicates have been defined to describe these height relations. These predicates are *isLocatedLower, isLocatedHigher*, and *isLocatedAtTheSameLevel*. As shown in Fig.4.15a, these predicates are originating from a primary landmark and pointing to the reference landmark to create an RDF statement representing the height relation between two landmarks.

In the helperNoder approach, as shown in Fig.4.15b, the *haveTheHeightRelation* predicate has been defined to link the helperNode to a literal that contains the height relation of the primary landmark to the reference landmark. According to the altitude information, the literal contains one of the following values, *higher, lower and same level* representing the height relation between two related landmarks.



(a) Binary approach **(b)** helperNode approach

Fig. 4.15: An illustration of the height relation between the post office and the cherry tree in the binary and helperNode approaches.

The subsequent code section represents the binary approach for modeling the height relation between two landmarks as shown in Fig.4.15a.

Listing 4.11: RDF-statements representing the binary approach for modeling height relations between the post office and the cherry tree.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:isLocatedLower rs:CherryTree_T02 ;
```

The subsequent code section represents the helperNode approach for modeling the height relation between two landmarks as shown in Fig.4.15b.

Listing 4.12: RDF-statements representing the helperNode approach for modeling the height relation between the post office and the cherry tree.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:towardsHelperNode  rs:B04T02 ;
rs:B04T02
            pr:fromHelperNode  rs:CherryTree_T02 ;
            pr:haveTheHeightRelation  "Lower" ;
```

4.5.3 *In the middle of* relation

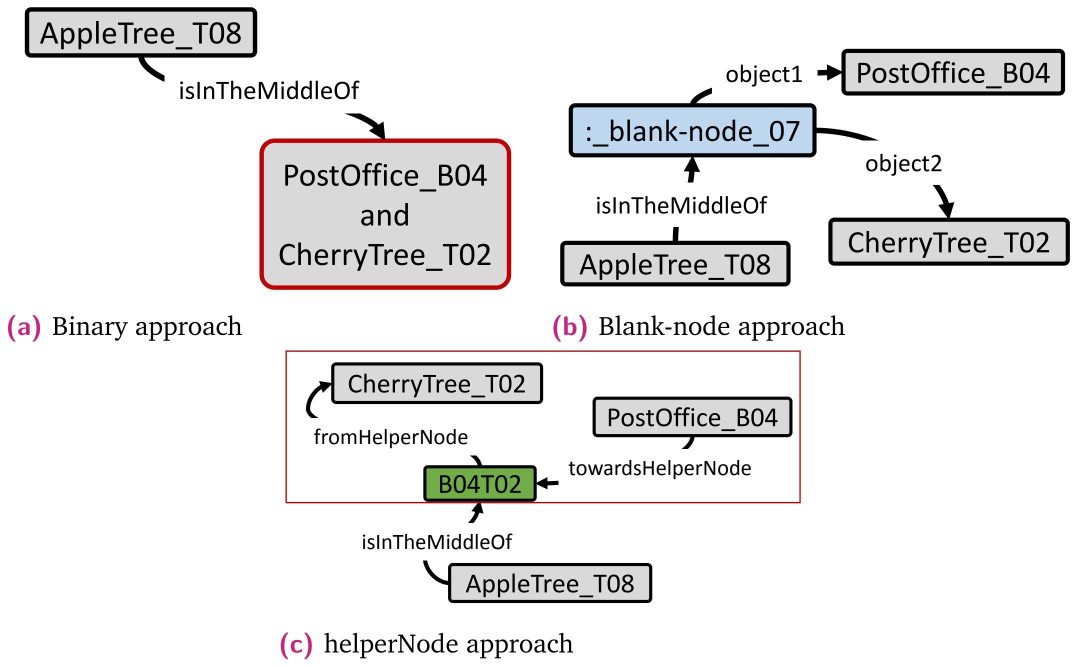
At each pose node, if AMOR detects three landmarks, it is estimated whether one of the detected landmarks is placed in the middle of the other two. To compensate for the gap between human and AMOR in the environment description the mathematical definition of the middle of has not been considered. Instead, just like a human in real life, AMOR describes a landmark in the middle of two others, if it is positioned in the area close to the center point of the two.

Due to the fact that three landmarks are involved in modeling a single relation that describes the middle of relation, this relation is considered a complex relation. The following three approaches have been used for modeling the middle of relation:

- **Binary approach:** the "middle of" relation consists of three individuals and to be able to use the binary approach for modeling this relation, they need to be reduced to two individuals. Therefore, a new individual has been created to represent the two reference landmarks. As shown in Fig.4.16a, the primary landmark is then linked to the newly created node with the *isInTheMiddleOf* predicate.

- **Blank-node approach:** In this approach, an anonymous blank-node has been added to the relation, to refer the relation to the two reference landmarks. As shown in Fig.4.16b, the *isInTheMiddleOf* predicate is created and originated from the primary landmark pointing to the blank-node. The blank-node is then referring the relation to the first reference landmark with the *object1* and the second reference landmark with the *object2* predicates.

- **HelperNode approach:** In this approach, as shown in Fig.4.16c, a helperNode is created to represent both reference landmarks. The *isInTheMiddleOf* predicate is then created and originated from the primary landmark pointing to the helperNode.



(a) Binary approach (b) Blank-node approach (c) helperNode approach

Fig. 4.16: An illustration of the middle of relation with different approaches.

The subsequent code section represents the binary approach for modeling the *middle of* relation, as shown in Fig.4.16a.

Listing 4.13: RDF-statements representing the binary approach for modeling the *isInTheMiddleOf* relation.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AppleTree_T08
            pr:isInTheMiddleOf rs:PostOffice_B04andCherryTree_T02 ;
```

The subsequent code section represents the blank-node approach for modeling the *isInTheMiddleOf* relation, as shown in Fig.4.16b.

Listing 4.14: RDF-statements describing the blank-node approach for modeling *isInTheMiddleOf* relation.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

```

```
rs:AppleTree_T08
              pr:isInTheMiddleOf   _:blank-node_07  ;
_:blank-node_07
              pr:object1   rs:CherryTree_T02  ;
              pr:object2   rs:PostOffice_B04  ;
```

The subsequent code section represents the helperNode approach for modeling the *isInTheMiddleOf* relation as shown in Fig.4.16c.

Listing 4.15: RDF-statements representing the helperNode approach for modeling *isInTheMiddleOf* relation.

```
@prefix  rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix  pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
              pr:towardsHelperNode   rs:B04T02  ;
rs:B04T02
              pr:fromHelperNode   rs:CherryTree_T02  ;

rs:AppleTree_T08
              pr:isInTheMiddleOf   rs:B04T02  ;
```

4.5.4 AMOR and the spatial prepositions

As a result of our biological makeup and the ways we understand and describe our environment, we as humans perceive our surrounding objects and their interrelations in a specific way. Despite language differences, almost all speakers use spatial prepositions to describe the spatial relations between objects. These prepositions describe the location of one object in respect to another object; examples are "right", "left", "behind" and "in front of" prepositions. Considering these prepositions describe the relation of one object to another, they are also called relational prepositions. One application of relational prepositions is an implicit expression of the point of view of the speaker, for instance, consider the following statement, "*the building is behind the tree*". In the mentioned statement one can implicitly indicate the point of view of the speaker. However, the relational prepositions can also be used differently by stating the point of view of the speaker explicitly, for instance, "*Looking from the factory the building is behind the tree*". In the second example, the point of view of the speaker is provided explicitly. In general, relational prepositions are highly dependent on the speaker's point of view (Brenda, 2014; Mohammed Saeed et al., 2018).

For modeling RDF statements representing relational prepositions between landmarks, the point of view of AMOR as the speaker is taken into consideration. While navigating, if AMOR perceives two landmarks, their Euclidean distance and their angle to the

AMOR is calculated by a scalar dot product to determine the relational preposition between them. For modeling these RDF statements, the following three different approaches have been employed.

- **Blank-node approach:** Three individuals are involved in creating a single preposition relation. These individuals are, the two related landmarks and a pose node in which they have been perceived. In a binary approach, only two individuals are allowed to model a single binary relation. Therefore, to overcome this problem, a predicate is created that represents the relational preposition between two related landmarks as well as the pose number in which the relation is valid. As shown in Fig.4.17a, the predicate is then originating from a primary landmark pointing to the reference landmark. However, as mentioned earlier, adding extra information to the predicate will lead to the creation of a huge number of predicates that results in increasing the size and complexity of the overall ontology.

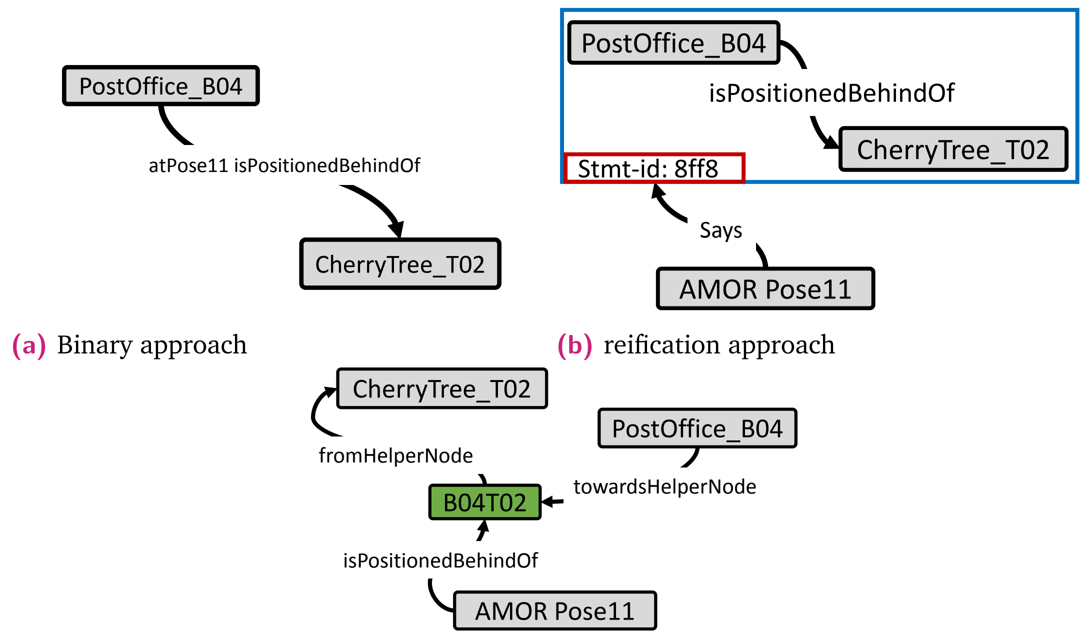
- **Reification approach:** In this approach, four different predicates have been defined to model RDF statements that describe the relationship between two related landmarks. Each of these statements is assigned with a unique id that is later used for the reification process. These predicates are *isPositionedBehindOf, isPositionedLeftOf, isPositionedInfrontOf*, and *isPositionedRightOf*. The AMOR-Pose node which had detected the related landmarks and observed their relation is then reified to its corresponding statement-id with the *says* predicate to indicate the point of view in which the statement is valid, (Mohammed Saeed et al., 2018); an example is shown in Fig.4.17b.

- **helperNode approach:** In this approach, as shown in Fig.4.17c, a helperNode is created to represent both reference landmarks. One of the relational preposition's predicate is then created and originates from the AMOR-Pose node which had detected the related landmarks and observed their relation pointing to the helperNode.

The subsequent code section represents the binary approach for modeling the prepositional relations between two landmarks, as shown in Fig.4.17a.

Listing 4.16: RDF-statements representing the binary approach for modeling prepositional relations between two landmarks relation.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:atPose11 isPositionedBehindOf rs:CherryTree_T02 ;
```



(a) Binary approach **(b)** reification approach **(c)** helperNode approach

Fig. 4.17: An illustration of the use of relational prepositions for modeling RDF-statements.

The subsequent code section represents the reification approach for modeling the prepositional relations between two landmarks, as shown in Fig.4.17b.

Listing 4.17: RDF-statements representing the reification approach for modeling the prepositional relations between two landmarks.

```
<urn:x-bnode:-5ac103e2:167b6223379:-7ffe>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement> .

<urn:x-bnode:-5ac103e2:167b6223379:-7ffe>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://www.eti.uni-siegen.de/ezls/ontology/resource/CityHall_B06> .

<urn:x-bnode:-5ac103e2:167b6223379:-7ffe>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://www.eti.uni-siegen.de/ezls/ontology/property/
    isPositionedBehindOf> .

<urn:x-bnode:-5ac103e2:167b6223379:-7ffe>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#object>
<http://www.eti.uni-siegen.de/ezls/ontology/BigMaple_T03> .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose3>
<http://www.eti.uni-siegen.de/ezls/ontology/resource/Says>
<urn:x-bnode:-5ac103e2:167b6223379:-7ffe> .
```

The subsequent code section represents the helperNode approach for modeling the prepositional relations between two landmarks, as shown in Fig.4.17c.

Listing 4.18: RDF-statements representing the helperNode approach for modeling the prepositional relations between two landmarks.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:PostOffice_B04
            pr:towardsHelperNode  rs:B04T02 ;
rs:B04T02
            pr:fromHelperNode  rs:CherryTree_T02 ;

rs:AMOR Pose11
            pr:isPositionedBehindOf  rs:B04T02 ;
```

4.6 Comparison between the blank-node and reification, binary and the helper node approaches

In section 4.5, different approaches have been utilized for modeling a number of RDF relations representing the inter-object relationship between detected landmarks of the environment. These approaches are the binary approach, blank-node approach, reification approach, and the proposed helperNode approach.

In the binary approach, the idea is to model inter-object relations in binary relations. However, in some cases, more than two nodes are involved in modeling a single inter-object relation; this raises the complexity and requires the elimination of one node by merging it with another node or in some case with the predicate of the relation. In the *hasDistance* relation, as shown in Fig.4.14a, the literal value representing the distance between two landmarks has been merged into the predicate, the same applies to the relations represented with relational prepositions. This approach of merging leads to the creation of a new predicate each time a new relation is created, this results in an increase in the size and complexity of querying such relations.

In the same way, the *isInTheMiddleOf* relation requires three individuals to model a single relation. To model *isInTheMiddleOf* relations in the binary approach requires the elimination of an individual by merging it to another individual; an example is shown in Fig.4.16a. However, in this approach of merging, the possibility of referencing the newly created individual with the real individuals that has been created from has not been provided; this raises the complexity and inconsistency of the overall ontology.

The blank-node approach has been employed for modeling a set of relations representing inter-object relations between landmarks; examples of this includes, the *hasDistance* and *isInTheMiddleOf* relations. Unlike the binary approach, the blank-node approach does not require elimination of nodes from the relation; it additionally, adds a new blank node to the relation that refers the relation itself into other nodes, examples of which are shown in Fig.4.15a and Fig.4.16b. However, as described in section 2.5.1.1, by applying the blank-node approach one encounters some drawbacks and disadvantages which have been addressed in works of (Hogan, 2015; Heath and Bizer, 2011; Gutierrez et al., 2004; Tzitzikas et al., 2012; Pichler et al., 2008; Käfer et al., 2013). Additionally, the blank-node approach can not be used to model relations represented with relational prepositions between landmarks.

Since the relations described with the relational prepositions are not mere facts rather than one's point of view, these relations require a meta-triple to provide additional information about the statement; detailed information has been provided in section 2.5.1.1. This work employs the reification approach for modeling such relations; examples, as shown in Fig.4.17b, are *isPositionedLeftOf*, *isPositionedRightOf*, *isPositionedBehindOf*, and *isPositionedInfrontOf*. The disadvantages of this approach have also been described in detail in section 2.5.1.1, and in the research of (Hayes, Patrick, 2004; Nguyen and Siberski, 2013; Nguyen et al., 2015; Groth et al., 2010; Sahoo et al., 2011; Callahan and Dumontier, 2013).

To address problems encountered with the binary, blank-node, and reification approaches this work has proposed the helperNode approach. As described in section 4.5, in this approach, a new node called the helperNode is created from the combination of codes of two related nodes and used to represent all relations between the two related nodes. As presented in the previous sections, the helperNode approach has been employed to model the following relations between related landmarks in the simulated environment.

1. *hasDistance*.
2. *hasTheHeightRelation*.
3. *isPositionedRightOf*.
4. *isPositionedLeftOf*.
5. *isPositionedBehindOf*.
6. *isPositionedInfrontOf*.

7. *isInTheMiddleOf*.

To evaluate and examine these different approaches an experimental navigation has been conducted. For proper evaluation and the possibility of comparing different approaches, three repositories in the AllegroGraph have been created. The first repository represented the binary approach and is called the *binary* repository. Since the blank-node and reification approaches are not able to model all given relations between related landmarks, they have been combined in a single repository called the *blanknodeAndReification* repository. The third repository represented the helperNode approach and is accordingly called the *helperNode* repository. Navigation of AMOR in its simulated environment for 43 seconds led to the creation of a number of RDF statements; Table 4.2 shows the number of predicates, nodes, reification statements, blank-nodes and helperNodes in each repository.

	binary	blanknodeAndReification	helperNode
AMOR-Pose	12	12	12
statements	397	1093	453
predicates	54	29	25
reification statements	0	18	0
blank-nodes	0	186	0
helperNodes	0	0	26

Tab. 4.2: Comparison between different approaches.

The information presented in Table 4.2 has been illustrated in a graph and shown in Fig.4.18.

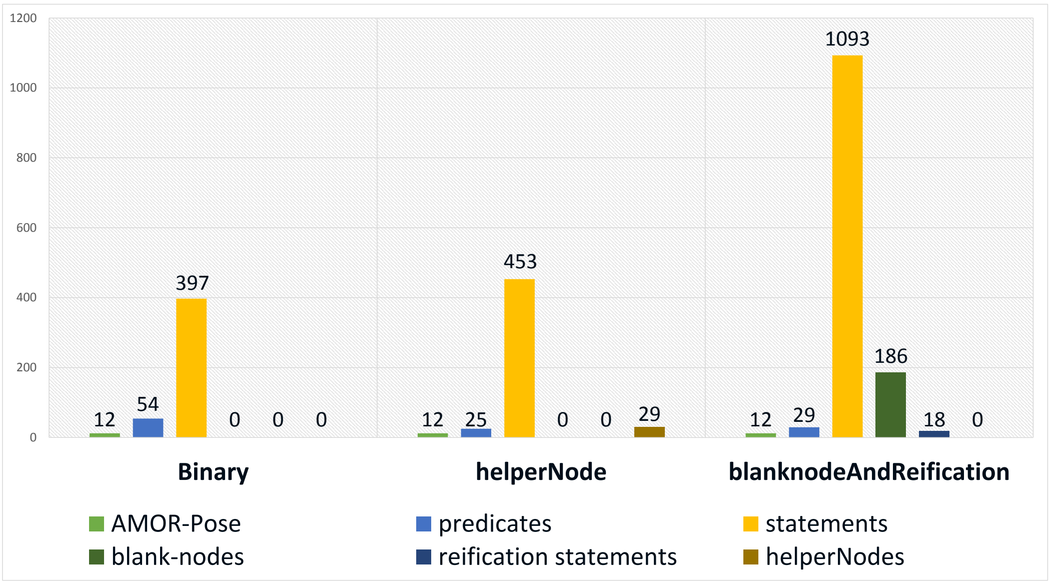


Fig. 4.18: An illustration of the information presented in Table 4.2.

4.7 Defining inverse relations

Jan is Anna's brother but does that mean Anna is Jan's sister? Hearing this first clause one can assume the second one. In daily life, humans often use inverse relations to extract information and comprehend meta-information which is implicit in sentences. In the semantic web, the RDFS provided the predicate *OWL:inverseOf* to enable ontologies to define inverse relations. When defining the inverse relation it is essential to notice that the position of the subject and object of the sentence changes. For instance, the *AppleTree isPositionedLeftOf CherryTree*, in the inverse sentence using the inverse relation *isPositionedRightOf*, the *CherryTree* becomes the subject and AppleTree becomes the object of the sentence. The inverse sentence is then as follows: *Cherrytree isPositionedRightOf AppleTree*.

This work utilizes the *OWL:inverseOf* predicate for defining a number of inverse relations to provide meta-information to the RDF statements created during navigation. Table 4.3 shows the inverse relations defined using the *OWL:inverseOf* predicate.

relation	inverse relation
hasPosition	positionOf
hasTime	timeOf
hasLocation	locationOf
hasDirection	directionOf
movesTo	movesFrom
hasYawAngleDerivative	yawAngleDerivativeOf
sees	hasBeenSeenBy
hasType	typeOf
withVicinityType	vicinityTypeOf
hasDistance	DistanceOf
hasValue	valueOf
locatedLower	locatedHigher
locatedHigher	locatedLower
hasTheHeightRelation	theHeightRelationOf
isPositionedRightOf	isPositionedLeftOf
isPositionedLeftOf	isPositionedRightOf
isPositionedBehindOf	isPositionedInfrontOf
isPositionedInfrontOf	isPositionedBehindOf

Tab. 4.3: A list of inverse relations defined in the RoboSemProc.

The subsequent code section is an example of using the *OWL:inverseOf* predicate for creating inverse relations for the relations mentioned in the Table 4.3.

Listing 4.19: RDF-statements representing an example of inverse relations created by using the *OWL:inverseOf* predicate.

```
@prefix owl:<http://www.w3.org/2002/07/owl#>.
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

pr:hasPosition
                owl:inverseOf  pr:positionOf ;
pr:hasTime
                owl:inverseOf  pr:timeOf ;
pr:hasLocation
                owl:inverseOf  pr:locationOf ;
pr:hasDirection
                owl:inverseOf  pr:directionOf ;
```

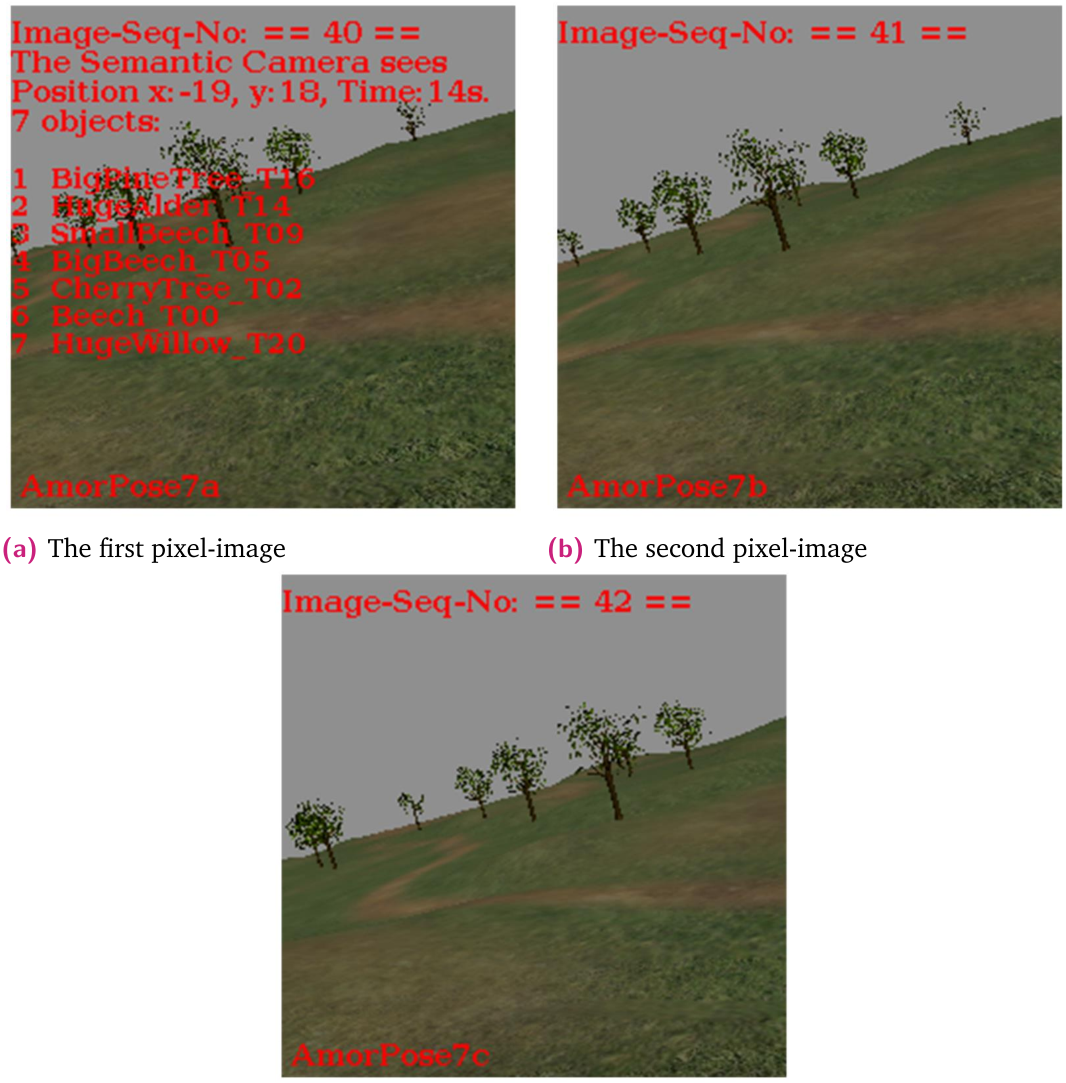
4.8 Sensor integration

AMOR has been equipped with a number of sensors; all mentioned RDF statements have been created with the sensor data of the pose sensors, and the semantic camera. Moreover, a number of RDF statements representing the inter-object relation between detected landmarks have also been created with the combination of sensor data of both the pose sensor and the semantic camera. Depending on the task and application different sensors can be equipped to AMOR. Therefore, it is essential to be able to integrate new sensors into the system and also semantically represent their data.

The environmental observation of AMOR has been based on the sensory data of the semantic camera, whereas the video camera outputs pixel-images and represents the visuals vision of AMOR at a given time.

Due to the sensitivity of the semantic camera, a landmark can be detected if only a small part of it has been placed in the field of view of the semantic camera, this information is hard to be detected by the vision of the video camera. To compensate the gap between these two cameras, at each pose that the semantic camera detects an object or a set of objects, three snapshots of the video camera are retrieved. The first snapshot is a pixel-image that has been recorded slightly before the output of the semantic camera, whereas the second and third pixel-images have been taken respectively at the same time, and slightly after detection of an object by the semantic camera. These images, as shown in Fig.4.19, have been named and annotated with their corresponding pose node followed by the letters a, b, c, representing the first, second and the third snapshots, respectively. The first snapshot, as shown in Fig.4.19a, has been annotated with the names of detected landmarks by the semantic camera.

The *seesTheImage* predicate is created and originated from the AMOR-Pose node pointing to the first pixel-image taken by the video camera. The *hasAdditionalConsecutiveImage_a* and *hasAdditionalConsecutiveImage_b* are predicates originating from the



(a) The first pixel-image (b) The second pixel-image

(c) The third pixel-image

Fig. 4.19: Three pixel-images captured by the video camera during navigation.

AMOR-Pose node pointing to its second and third pixel-images, accordingly. An RDF example is presented in the subsequent code section.

Listing 4.20: RDF-statements representing an example of statements created by the sensor data of the video camera.

```

@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .

rs:AMOR Pose7
            pr:seesTheImage   "AMORPose7a" ;
            pr:hasAdditionalConsecutiveImage_a   "AMORPose7b" ;
            pr:hasAdditionalConsecutiveImage_b   "AMORPose7c" ;
```

4.9 Navigation and real-time ontology instantiation

A set of sensor data provide AMOR with its exact position, orientation, time, and moreover it presents a semantic and vision of AMOR's environment. These sensory inputs have been utilized to create a set of RDF statements that describe the tour of AMOR in a semantic manner. While navigating, these RDF statements are created in real-time and used to instantiate the AMOR core ontology. This ontology, as described in section 4.2, and shown in Fig.4.4, includes a set of classes and subclasses designed to represent the tour of AMOR as well as its vision and understanding of its environment. To populate the AMOR core ontology the following classes of the ontology have been instantiated and linked to the real-time created RDF statements with *ObjectProperty:isA*. An example is shown in Fig.4.20.

- **EZLS:Pose** The Pose class of the AMOR core ontology has been populated with all AMOR-Pose nodes.
- **CORA:TimePoint**: This class has been instantiated with time properties of the AMOR-Pose nodes.
- **EZLS:Location**: The location class has been instantiated with the location of all AMOR-Pose nodes and also location of all detected landmarks.
- **EZLS:Position**: This class has been instantiated with the position information of all AMOR-Pose nodes and also the position of all detected landmarks.
- **EZLS:Tree**: All detected landmarks of type tree have been associated to the Tree class of the AMOR core ontology.
- **Building**: All detected landmarks of type building have been associated to the Building class of the AMOR core ontology.

The subsequent code section demonstrates an example shown in Fig.4.20, which demonstrates the instantiating of the AMOR core ontology with RDF statements of a single AMOR-Pose node.

Listing 4.21: An example of the instantiating of the AMOR core ontology with RDF statements of a single AMOR-Pose node.

```
@prefix rs:<http://www.eti.uni-siegen.de/ezls/ontology/resource> .
@prefix pr:<http://www.eti.uni-siegen.de/ezls/ontology/property> .
@prefix ins:<http://www.w3.org/ns/lemon/synsem#isA >.

rs:AmorPose9
```

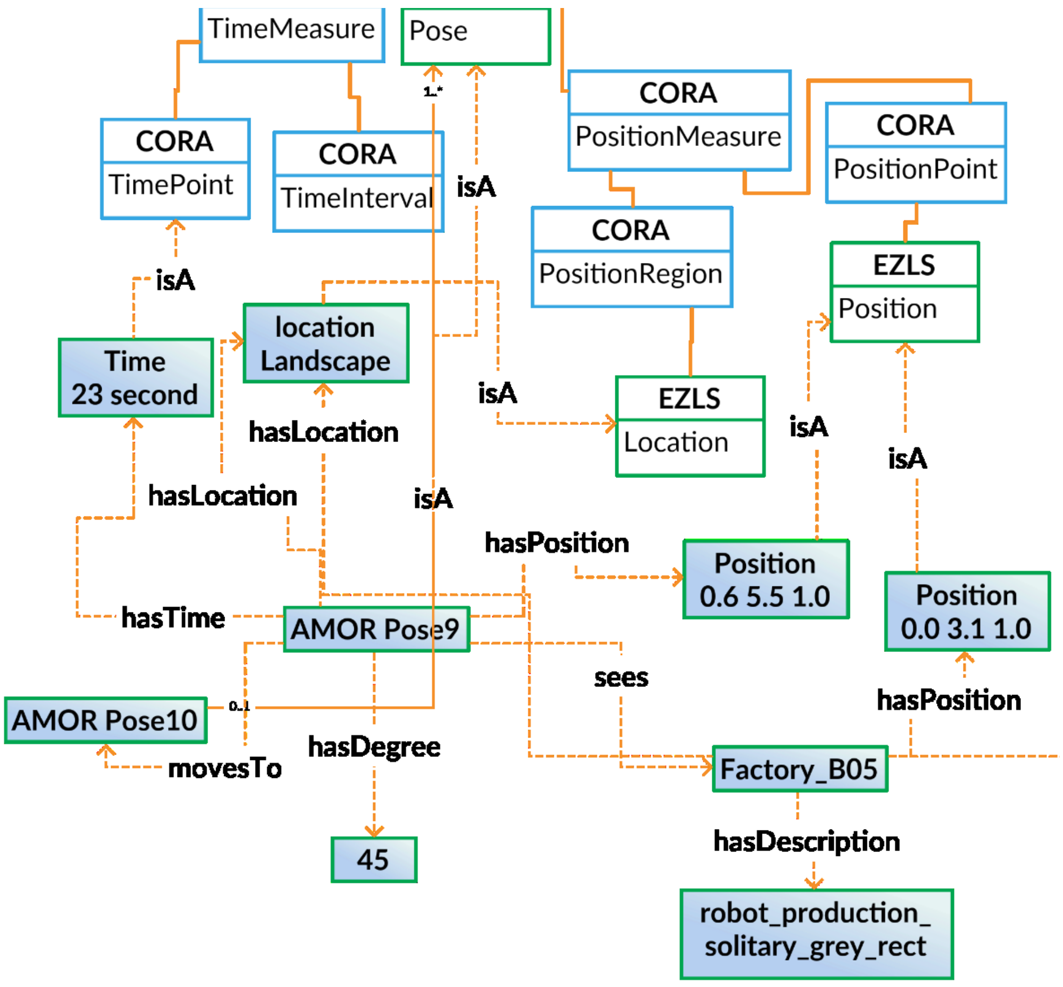


Fig. 4.20: A demonstration of the AMOR core ontology instantiation with a set of RDF statements created during navigation.

```
            inst:isA  rs:Pose
            pr:hasTime  rs:Time:  23  Seconds  ;
            pr:hasPosition  rs:Position:  0.0  3.1  1.0  ;
            pr:hasLocation  rs:Location:  landscape;
             pr:hasDirection  "45"  ;
            pr:movesTo  rs:AMORPose10;
             pr:hasYawAngleDerivative  rs:"increasingYawAngle";
            pr:sees  rs:Factory_B05;

rs:Factory_B05
            pr:hasType  rs:Building  ;
            pr:hasDescription  "robot_production_solitary_grey_rect"  ;
            pr:hasPosition  rs:Position:  0.6  5.5  1.0  ;
            pr:hasLocation  rs:Location:  landscape;
            pr:withVicinityType  "close  proximity";

rs:Time:  23  Seconds
            inst:isA  rs:TimePoint;

```

```
rs:Position: 0.0 3.1 1.0
               inst:isA rs:Position;

rs:Position: 0.6 5.5 1.0
               inst:isA rs:Position;

rs:Location: landscape
               inst:isA rs:Location;

rs:AMORPose10
               inst:isA rs:Pose;
```

4.10 An external Environment Ontology *"ENVO"* integration

One of the key reasons for the popularity of ontologies is their ability to be linked and reused by other ontologies. Linking ontologies enables the reuse of their semantic information by different agents; open linked data is a good example of this. This research utilizes this great feature and extends the AMOR core ontology with an external ontology to provide a complete definition of building type landmarks. In this way, AMOR is provided with the complete definitions of buildings. Moreover, this is an indication that shows the possibility of extending the AMOR core ontology with other ontologies, depending on the application and need in the future.

The ontology that has been chosen for the extension of the AMOR core ontology is called environment ontology (ENVO[2]). The ENVO ontology is a general ontology that describes environments and its related entities. It inter-operates with other ontologies and consists of definitions for 6909 terms. It includes a set of classes that best define buildings, their types, parts, and properties (European Molecular Biology, 2018). To avoid complexity the building-related part of the ENVO ontology has been extracted and replaced with the building class of the AMOR core ontology. The extracted part of the ENVO ontology consists of 1706 statements. The basic classes of the ENVO ontology representing buildings is shown in Fig.4.21.

To extend the AMOR core ontology, as shown in Fig.4.22, the building class of the AMOR core ontology has been replaced with the building class of the ENVO ontology, which internally has its own set of classes and properties that define buildings.

[2]https://www.ebi.ac.uk/ols/ontologies/envo

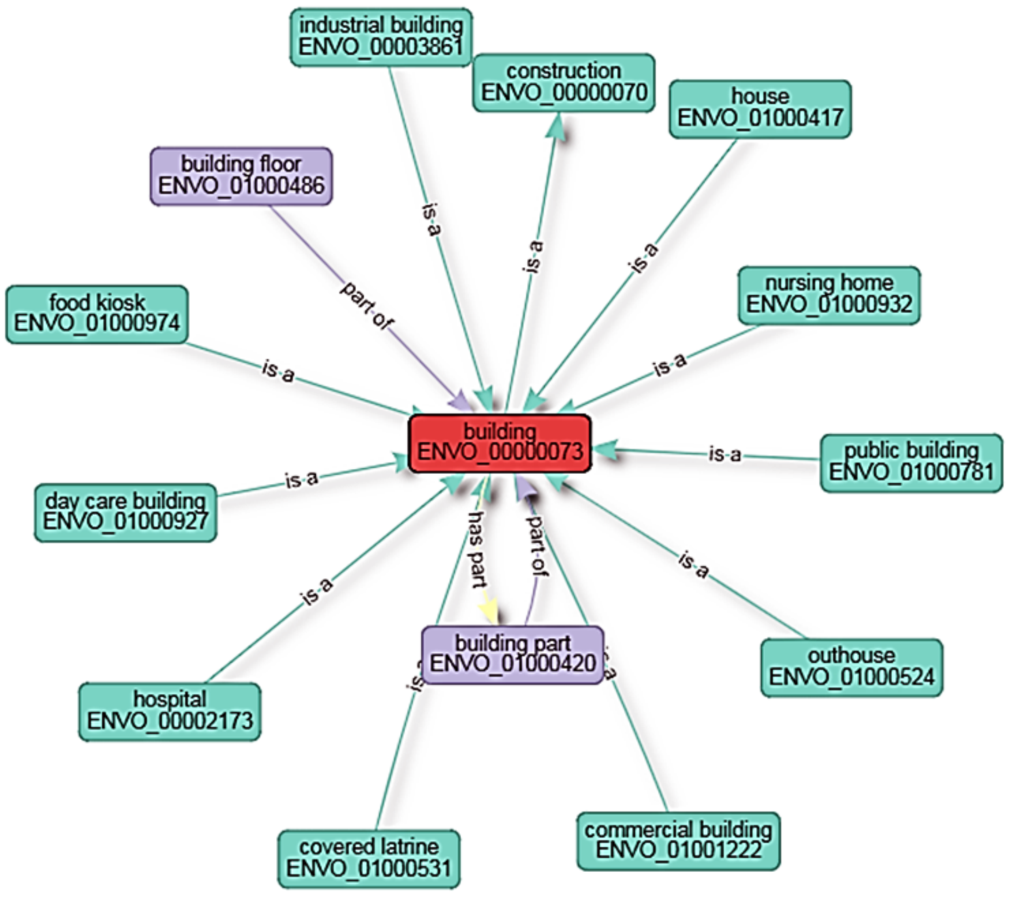

industrial building ENVO_00003861
construction ENVO_00000070
house ENVO_01000417
building floor ENVO_01000486
nursing home ENVO_01000932
food kiosk ENVO_01000974
building ENVO_00000073
public building ENVO_01000781
day care building ENVO_01000927
building part ENVO_01000420
outhouse ENVO_01000524
hospital ENVO_00002173
covered latrine ENVO_01000531
commercial building ENVO_01001222
is-a
part-of
has part

Fig. 4.21: The visualization of the main ENVO ontology classes that represent buildings. The graph has been generated by the Ontology Lookup Service, visualization tool (European Molecular Biology, 2018).

4.11 Semantic description and communication

In previous sections, the process of modeling semantic information from sensor data and instantiating the AMOR core ontology during navigation has been described. This section describes the use of this ontology for a communication in a natural language.

One way to use ontologies for natural language communication is the use of QA. As mentioned in section 2.8, the QA defines techniques and methods for intelligent agents to answer human questions in a natural language. It can also be referred to as the process of translating questions asked by users and mapping them into their semantics with a natural language representation and providing a valid answer (Trischler et al., 2016). Detailed information about QA is provided in section 2.8.

As described in section 2.8, in the last decade, several studies have been conducted, and a number of QA systems have been proposed, examples of which include OpenE-

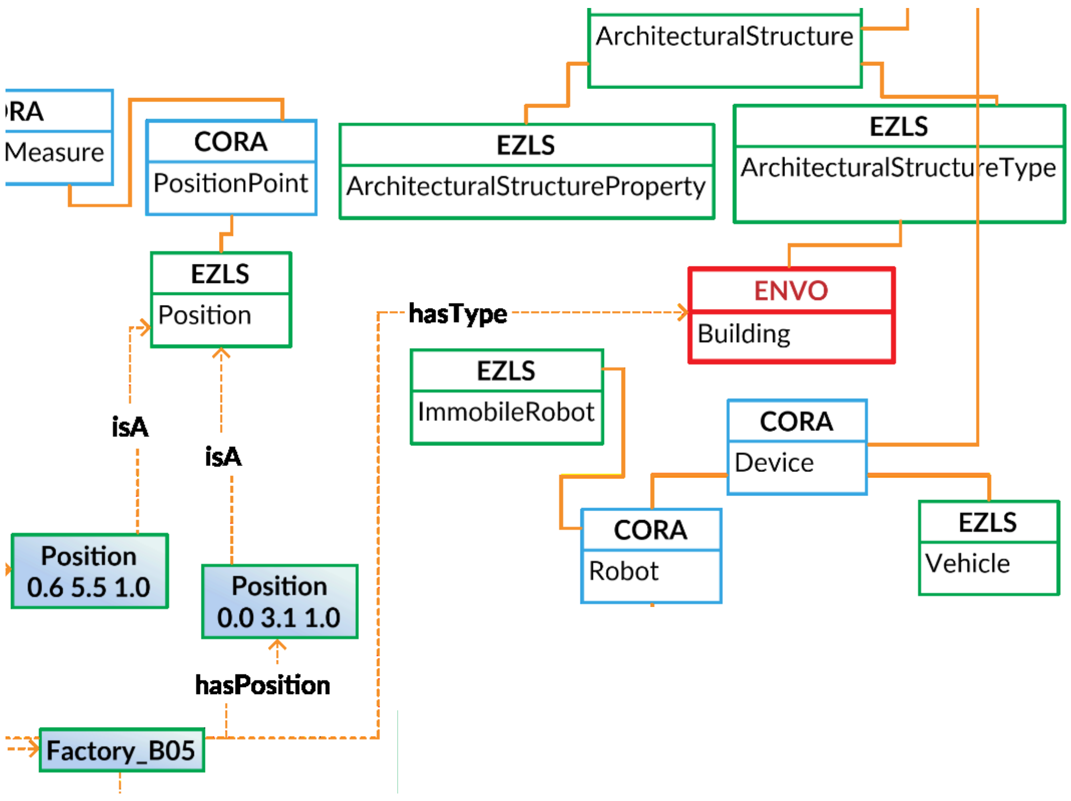


Fig. 4.22: A demonstration of the AMOR core ontology linked with the ENVO building class, to provide definitions to buildings, the *ENVO:Building* class has been highlighted.

phyra, Open Advancement of Question Answering (OAQA), OpenQA, DeepQA, and Yet anOther Deep Answering pipeline Question Answering (YodaQA) system. Among the mentioned systems, the YodaQA, which has been inspired by the IBM Watson's framework DeepQA, is the most promising system and is suitable for both structured and unstructured data sources. With the correct answering rate of about 80% it is considered as one of the most accurate QA system available. It provides a modular and open source system that allows the integration of different knowledge bases. Therefore, this work utilizes the YodaQA system as a QA system for the natural language communication with AMOR.

4.11.1 Domain adaptation

YodaQA is an open source QA system that enables adaptation of new domains. To be able to adapt the AMOR core ontology to the YodaQA system and use it for natural language communication the following new repositories have been created:

- *infobox_property_definitions_en*: For the YodaQA system to be able to identify properties or the so-called predicates of RDF-statements, they all need to be associated with the Property class of the RDFS. Hence, the system has been designed in a way that detects all predicates and annotates them with additional information and assigns them to the Property class. YodaQA in general searches through labels, therefore for the information to be available to the system, they need to be associated with labels. Hence, the next required step is to add a label to each property. The properties and classes of the W3C shown in Table 4.4 have been employed to assign each predicate to the Property class and also for annotating each property with a label.

classes	predicate
Type	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
Property	<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>
label	<http://www.w3.org/2000/01/rdf-schema#label>

Tab. 4.4: The classes and properties of the W3C used for property definition and labeling.

The subsequent code section demonstrates an example of the property definition and labeling.

Listing 4.22: RDF-statements demonstrating the process of property definition and labeling.

```
<http://www.eti.uni-siegen.de/ezls/ontology/property/movesTo>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
          <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<http://www.eti.uni-siegen.de/ezls/ontology/property/movesTo>
        <http://www.w3.org/2000/01/rdf-schema#label> "movesTo" .

<http://www.eti.uni-siegen.de/ezls/ontology/property/location>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
          <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<http://www.eti.uni-siegen.de/ezls/ontology/property/location>
        <http://www.w3.org/2000/01/rdf-schema#label> "location" .

<http://www.eti.uni-siegen.de/ezls/ontology/property/direction>
     <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property> .
<http://www.eti.uni-siegen.de/ezls/ontology/property/direction>
        <http://www.w3.org/2000/01/rdf-schema#label> "direction" .
```

After finalizing the process of property definition and labeling, the resulting RDF-statements are collected and stored in the *infobox_property_definitions_en* repository to be later used by the YodaQA pipelines.

- *labels_en*: As mentioned earlier, to make information available to the YodaQA for natural language communication they all need to be enriched with a label. Despite their functionality in the YodaQA system, labeling also helps to simplify and provide more user-friendly names to each resource. Previously, the process of labeling properties has been described. After labeling properties it is necessary to provide a label to each resource. To do so, the same label property that has been used for the labeling properties has been employed to label all resources.

  The subsequent code section demonstrates an example of labeling resources.

  Listing 4.23: RDF-statements demonstrating the process of resource labeling.

```
<http://www.eti.uni-siegen.de/ezls/ontology/resource/HugeOak_T23>
<http://www.w3.org/2000/01/rdf-schema#label> "HugeOak_T23" .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose8>
<http://www.w3.org/2000/01/rdf-schema#label> "AmorPose8" .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/Museum_B07>
<http://www.w3.org/2000/01/rdf-schema#label> "Museum_B07" .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/Church_B09>
<http://www.w3.org/2000/01/rdf-schema#label> "Church_B09" .
```

  RDF-statements representing the resource labeling have been collected and stored in the *labels_en* repository, to be later used by the YodaQA system for the natural language communication process.

- *page_ids_en* and *redirects_transitive_en*:

  YodaQA can process structured and unstructured data sources. To provide detailed information about the answer, YodaQA provides the functionality for mapping the final answer to its corresponding Wikipedia page. Therefore each resource in the DBpedia[3] ontology used by YodaQA has been annotated with its corresponding Wikipedia page-id and wikiPageRedirects properties that has been used to redirect the resource to its corresponding Wikipedia page-id.

  This functionality has been applied to this work; therefore the following classes and proptries shown in Table 4.5 have been used to enrich each resource in the AMOR core ontology with a set of RDF-statements representing their wikiPageRedirects property and also their page-ids.

[3]https://wiki.dbpedia.org/

classes	predicate
WikiPageID	<http://dbpedia.org/ontology/wikiPageID>
Int	<http://www.w3.org/2001/XMLSchema#int>
WikiPageRedirects	<http://dbpedia.org/ontology/wikiPageRedirects>

Tab. 4.5: The classes and properties used for defining page-ids.

Two repositories namely *page_ids_en* and *redirects_transitive_en* have been created to store RDF statements representing page-ids and wikiPageRedirects accordingly. For this work, and more precisely for the demonstration of this functionality the Wikipedia page of the University of Siegen has been used. Hence, each resource has been annotated with the page-id: 12695897 which represents the University of Siegen. This information can be customized and used to link the resource to different pages depending on the need and application in the future.

The subsequent code sections demonstrate the use of the properties mentioned above for modeling RDF statements representing the Wikipedia redirect page and page ids to each resource.

Listing 4.24: RDF-statements code demonstrating the process of modeling RDF statements with the *wikiPageRedirects* property.

```
<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose1>
   <http://dbpedia.org/ontology/wikiPageRedirects>
<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose1> .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose2>
   <http://dbpedia.org/ontology/wikiPageRedirects>
<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose2> .
```

Listing 4.25: RDF-statements demonstrating the process of annotating RDF statements with their Wikipedia page-id.

```
<http://www.eti.uni-siegen.de/ezls/ontology/resource/AmorPose1>
<http://dbpedia.org/ontology/wikiPageID>
            "12695897"^^<http://www.w3.org/2001/XMLSchema#int> .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/CityHall_B06
    >
<http://dbpedia.org/ontology/wikiPageID>
            "12695897"^^<http://www.w3.org/2001/XMLSchema#int> .

<http://www.eti.uni-siegen.de/ezls/ontology/resource/B06T03>
<http://dbpedia.org/ontology/wikiPageID>
            "12695897"^^<http://www.w3.org/2001/XMLSchema#int> .
```

4.11.2 Apache Jena Fuseki and label lookup service

In the previous section, the process of resource labeling and annotating with the wikiPageRedirects property and wikiPageID has been described in which different repositories have been created to collect this information. To make use of this information, YodaQA utilizes the label-lookup[4] service motivated by (Bast and Haussmann, 2015; Spitkovsky and Chang, 2012; Yao, 2015). In the label-lookup service, all resources with their labels, and redirect page-ids are collected and presented in a single file. The idea is to allow remote operation with different SPARQL endpoints, and it is also to maintain memory.

This work implements the label-lookup service and creates a single lookup list to represent all resources. The list is created from the combination of the RDF statements stored in the *labels_en*, *redirects_transitive_en* and the *page_ids_en* repositories. After creating the updated label-lookup list, a simple python script runs the service and can be accessed over *http://localhost:5000/*. YodaQA uses this API to examine the availability of resources and also to retrieve the Wikipedia page-id of any desired resource. Table 4.6 shows the structure of an example label-lookup list.

resource	label	page-id	status
Alder_T11	Alder_T11	12695897	1
AmorPose10	AmorPose10	12695897	1
AmorPose9	AmorPose9	12695897	1
AppleTree_T08	AppleTree_T08	12695897	1
B01B03	B01B03	12695897	1

Tab. 4.6: The structure example of the label-lookup list.

As shown in Table 4.6, the label-lookup list provides the label and page-id of each resource. Moreover, this list provides a status value that indicates the consistency and validity of the resource. The value 1 indicates the correct status whereas the value 0 represents an incorrect status of the resource, any resource marked with the status 0 is not valid and is therefore eliminated from the search results.

As described, the *labels_en*, *redirects_transitive_en* and the *page_ids_en* repositories have been utilized for the creation of a label-lookup list.

YodaQA uses the Apache Jena Fuseki[5] as its SPARQL endpoint server, and therefore, the ontology and additional RDF statements presented in different repositories have to be uploaded and hosted by the Apache Jena server. The required repositories for this

[4]https://github.com/brmson/label-lookup/

[5]https://jena.apache.org/

purpose are, *labels_en*, and the *page_ids_en*, *infobox_property_definitions_en*, as well as the AMOR core ontology. More details are presented in the chapter that follows.

Results and evaluation 5

Innovation requires an experimental mindset.

— Denise Morrison
(global businesses leader.)

5.1 Experimental navigation, exploring the city and the landscape

In the previous chapters, the different phases of the RoboSemProc, starting from simulating AMOR and its environment, object annotation and pre-processing of the sensory data as well as ontology design and RDF modeling have been examined along with ontology instantiation and integration, and finally, the application of the system for natural language communication has been described. As discussed, the RoboSemProc operates in combination with different software systems; this can be seen in its user interface. As shown in Fig.5.1, several buttons on the left-hand side of the user interface of the RoboSemProc enable users to run different services and to control the flow of data in the system. The different tabs in the middle are providing users with detailed information about different phases of the system, such as outputting sensory data and asserted RDF statements. Two windows on the right-hand side provide the live view and snapshot of the captured images.

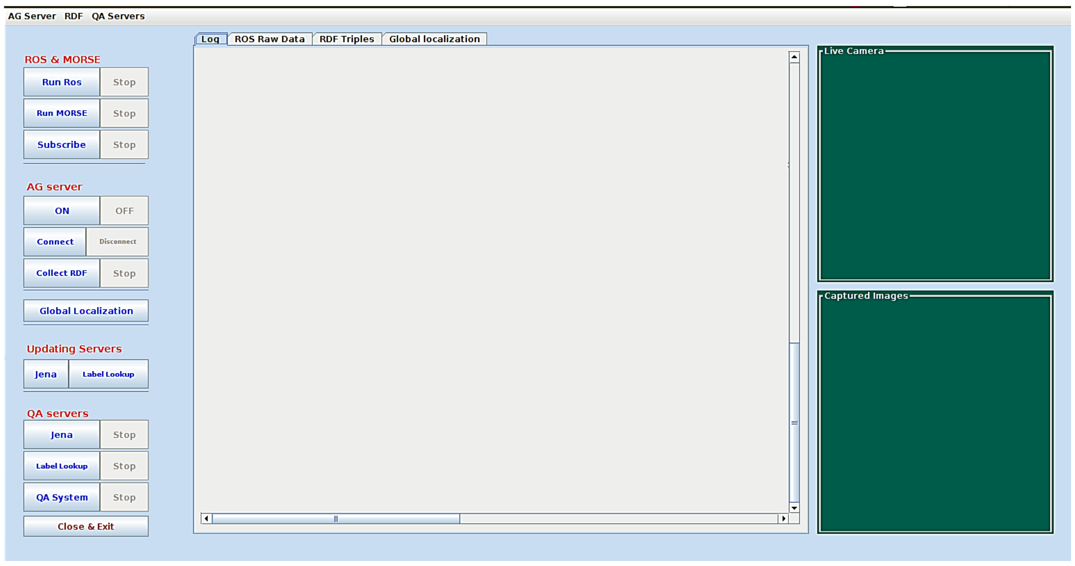
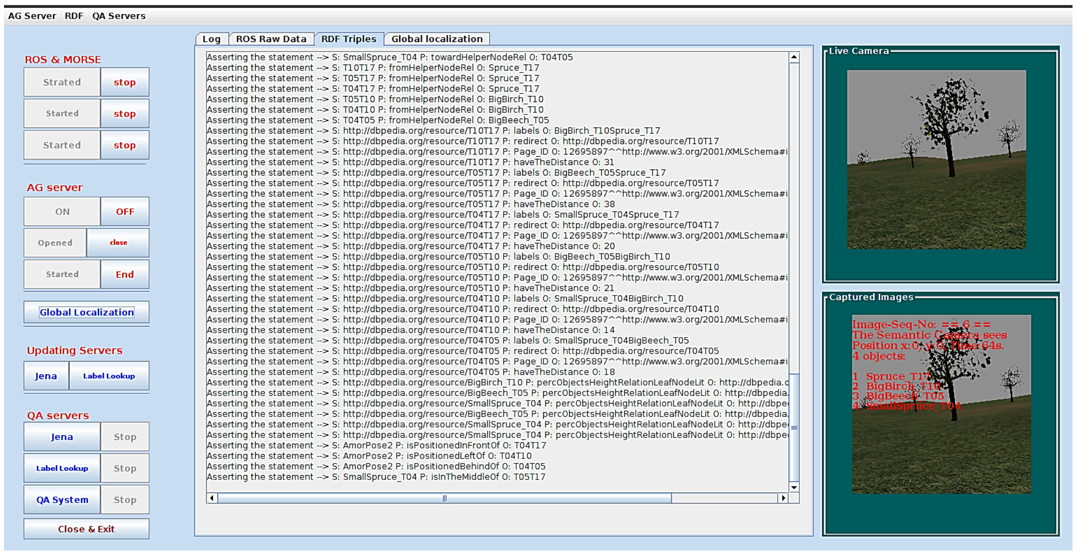


Fig. 5.1: The user interface of the RoboSemProc. As can be seen, the control buttons on the left-hand side of the user interface are categorized into four different sections; ROS and MORSE section, AG section, updating servers and run servers. The detailed information about each section is provided in the coming sections of this chapter.

The rest of this chapter presents the results of a sample tour of AMOR in its environment. As can be seen in Fig.5.2, to conduct a tour and populate the AMOR core ontology with the collected RDF statements the following services need to be running and active:

- ROS
- MORSE
- Subscription to the listener and talker of ROS-nodes
- AllegroGraph server
- A connection to the AllegroGraph
- Data collection

The experimental exploration of AMOR in its environment for 467 seconds resulted in the creation of a number of RDF statements which have been collected and stored in various repositories of the AllegroGraph. Table 5.1 shows the number of statements and predicate collected in each repository.



```
Asserting the statement --> S: SmallSpruce_T04 P: towardHelperNodeRel O: T04T05
Asserting the statement --> S: T10T17 P: fromHelperNodeRel O: Spruce_T17
Asserting the statement --> S: T05T17 P: fromHelperNodeRel O: Spruce_T17
Asserting the statement --> S: T04T17 P: fromHelperNodeRel O: Spruce_T17
Asserting the statement --> S: T05T10 P: fromHelperNodeRel O: BigBirch_T10
Asserting the statement --> S: T04T10 P: fromHelperNodeRel O: BigBirch_T10
Asserting the statement --> S: T04T05 P: fromHelperNodeRel O: BigBeech_T05
Asserting the statement --> S: http://dbpedia.org/resource/T10T17 P: labels O: BigBirch_T10Spruce_T17
Asserting the statement --> S: http://dbpedia.org/resource/T10T17 P: redirect O: http://dbpedia.org/resource/T10T17
Asserting the statement --> S: http://dbpedia.org/resource/T10T17 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T10T17 P: haveTheDistance O: 31
Asserting the statement --> S: http://dbpedia.org/resource/T05T17 P: labels O: BigBeech_T05Spruce_T17
Asserting the statement --> S: http://dbpedia.org/resource/T05T17 P: redirect O: http://dbpedia.org/resource/T05T17
Asserting the statement --> S: http://dbpedia.org/resource/T05T17 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T05T17 P: haveTheDistance O: 38
Asserting the statement --> S: http://dbpedia.org/resource/T04T17 P: labels O: SmallSpruce_T04Spruce_T17
Asserting the statement --> S: http://dbpedia.org/resource/T04T17 P: redirect O: http://dbpedia.org/resource/T04T17
Asserting the statement --> S: http://dbpedia.org/resource/T04T17 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T04T17 P: haveTheDistance O: 20
Asserting the statement --> S: http://dbpedia.org/resource/T05T10 P: labels O: BigBeech_T05BigBirch_T10
Asserting the statement --> S: http://dbpedia.org/resource/T05T10 P: redirect O: http://dbpedia.org/resource/T05T10
Asserting the statement --> S: http://dbpedia.org/resource/T05T10 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T05T10 P: haveTheDistance O: 21
Asserting the statement --> S: http://dbpedia.org/resource/T04T10 P: labels O: SmallSpruce_T04BigBirch_T10
Asserting the statement --> S: http://dbpedia.org/resource/T04T10 P: redirect O: http://dbpedia.org/resource/T04T10
Asserting the statement --> S: http://dbpedia.org/resource/T04T10 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T04T10 P: haveTheDistance O: 14
Asserting the statement --> S: http://dbpedia.org/resource/T04T05 P: labels O: SmallSpruce_T04BigBeech_T05
Asserting the statement --> S: http://dbpedia.org/resource/T04T05 P: redirect O: http://dbpedia.org/resource/T04T05
Asserting the statement --> S: http://dbpedia.org/resource/T04T05 P: Page_ID O: 12695897^^http://www.w3.org/2001/XMLSchema#i
Asserting the statement --> S: http://dbpedia.org/resource/T04T05 P: haveTheDistance O: 18
Asserting the statement --> S: http://dbpedia.org/resource/BigBirch_T10 P: percObjectsHeightRelationLeafNodeLit O: http://dbpedia.c
Asserting the statement --> S: http://dbpedia.org/resource/BigBeech_T05 P: percObjectsHeightRelationLeafNodeLit O: http://dbpedia.
Asserting the statement --> S: http://dbpedia.org/resource/SmallSpruce_T04 P: percObjectsHeightRelationLeafNodeLit O: http://dbpe
Asserting the statement --> S: http://dbpedia.org/resource/BigBeech_T05 P: percObjectsHeightRelationLeafNodeLit O: http://dbpedia.
Asserting the statement --> S: http://dbpedia.org/resource/SmallSpruce_T04 P: percObjectsHeightRelationLeafNodeLit O: http://dbpe
Asserting the statement --> S: http://dbpedia.org/resource/SmallSpruce_T04 P: percObjectsHeightRelationLeafNodeLit O: http://dbpe
Asserting the statement --> S: AmorPose2 P: isPositionedInFrontOf O: T04T17
Asserting the statement --> S: AmorPose2 P: isPositionedLeftOf O: T04T10
Asserting the statement --> S: AmorPose2 P: isPositionedBehindOf O: T04T05
Asserting the statement --> S: SmallSpruce_T04 P: isInTheMiddleOf O: T05T17
```

```
Image-Seq-No: == 6 ==
The Semantic Camera sees
Position x:0, y:0, Time:64s.
4 objects:

1  Spruce_T17
2  BigBirch_T10
3  BigBeech_T05
4  SmallSpruce_T04
```

Fig. 5.2: The user interface of the RoboSemProc in action.

Repository	RDF statements	RDF predicates
binary	4.371	285
blanknodeAndReification	14.757	79
helperNode	4.824	76
infobox_property_definition_en	7190	2
labels_en	1.866	1
page_ids_en	1.335	1
redirects_transitive_en	1.866	1

Tab. 5.1: The number of statements and predicates collected from exploring for 467 seconds.

For the described experimental exploration of AMOR in its environment for 467 seconds a number of RDF statements have been collected and stored in various repositories of the AllegroGraph. Table 5.2 shows the amount of memory size used for storing these statements and predicates collected in each repository. As can be seen in Table 5.2, the total amount of the memory size used for this experimental exploration in a dense environment was 28,888 KB. Considering this information, the approximate memory size required for one hour of navigation is 222.691 MB.

Repository	Memory size
binary	695 KB
blanknodeAndReification	1,817 KB
helperNode	712 KB
infobox_property_definition_en	1,221 KB
labels_en	214 KB
page_ids_en	255 KB
redirects_transitive_en	218 KB
Pixel-images	23,856 KB
Total amount of memory used	28,888 KB

Tab. 5.2: The the memory size used by the the mention exploration.

5.2 Visualization of the AMOR core ontology

Visualization of information, the presentation of data in a graphic format, enables decision-makers and owners of information to receive a visual analysis of their data; this makes it easier for a human to understand the difficult concepts and identify new patterns. Furthermore, it has long been proven that humans understand and memorize visual information easier. Doing different memory techniques, like the memory palace (Legge et al., 2012) and the journey method (Godwin-Jones, 2010), that are based on graphical memory are good examples of the importance of visualization. In the human mind, viewing and understanding a painting is quicker and easier than reading a page of a book. In comparison to tables, graphs are analyzed faster. The simplicity and efficiency of RDF over relational databases are not only premised on its machine understandability, human understandability is another advantage of RDF which is provided by its structure and visualization. In the following, the Gruff, which is the visualization tool of AllegroGraph has been used to demonstrate the collected RDF statements in RDF-graphs. Visualization of the data collected by the mentioned experimental navigation is demonstrated as an RDF-graph, and it is shown in Fig.5.3, and 5.4.

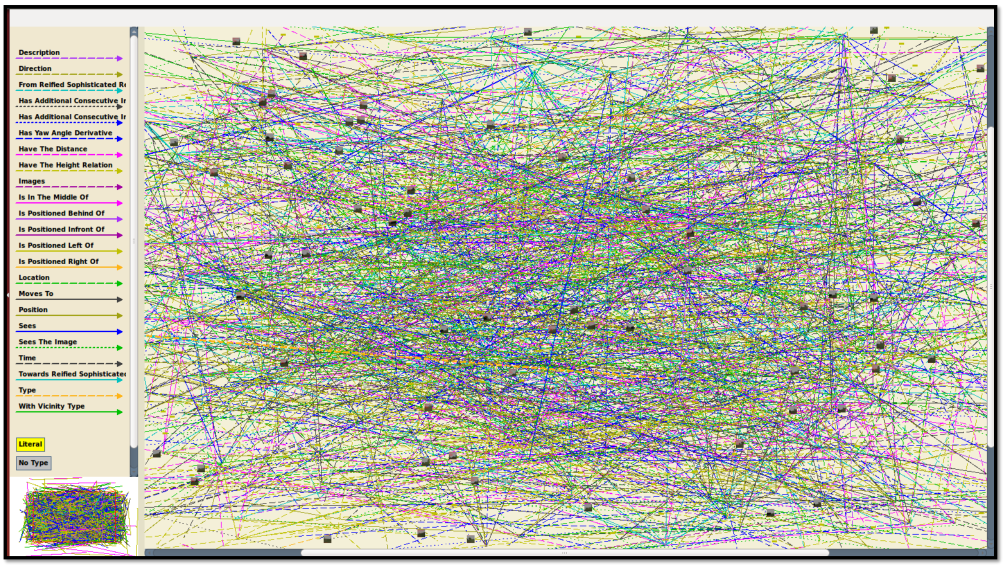


Fig. 5.3: Visualization of triples in Gruff. On the left-hand side, the relations connecting nodes are presented. Each colored line in the below graph represents a relationship that connects two nodes.

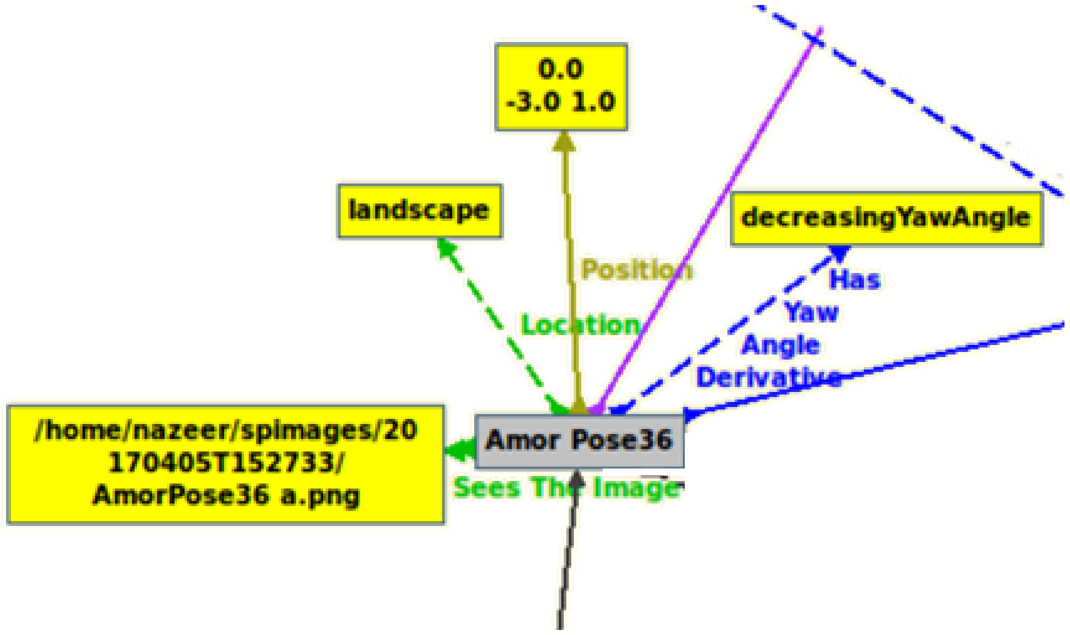


Fig. 5.4: RDF-Graph showing a closer look at the RDF-Graph shown in Fig.5.3.

5.3 Property definition, labeling and page-ids

While exploring the city and the landscape, several RDF-statements have been created to represent the tour of AMOR. Moreover, the semantic and video camera of AMOR provided information about detected landmarks that also resulted in the creation of additional RDF-statements, representing the explored environment. While creating these statements, a number of predicates have been used to connect resources to others or their corresponding values. During the process of property definition and labeling each of these predicates are associated to the Property class and also annotated with a label. As shown in Fig.5.5, this information has been collected and stored in *infobox_property_definition_en* repository and later used with the YodaQA system for natural language communication. Just like predicates all resources have also been annotated with labels; this enables the YodaQA search engine to find the desired resource. The *label_en* repository has been created to collect these RDF-statements. A visualization demonstration of RDF-statements stored in the *label_en* is given in Fig.5.6. Additionally, each resource is also provided with a *wikiPageID* and also *wikiPageRedirects* properties. This information is collected in *labels_en, page_ids_en* and *redirects_transitive_en* respectively. Fig.5.7, and Fig.5.8 show the visualization of the content of the *page_ids_en* and *redirects_transitive_en accordingly*.

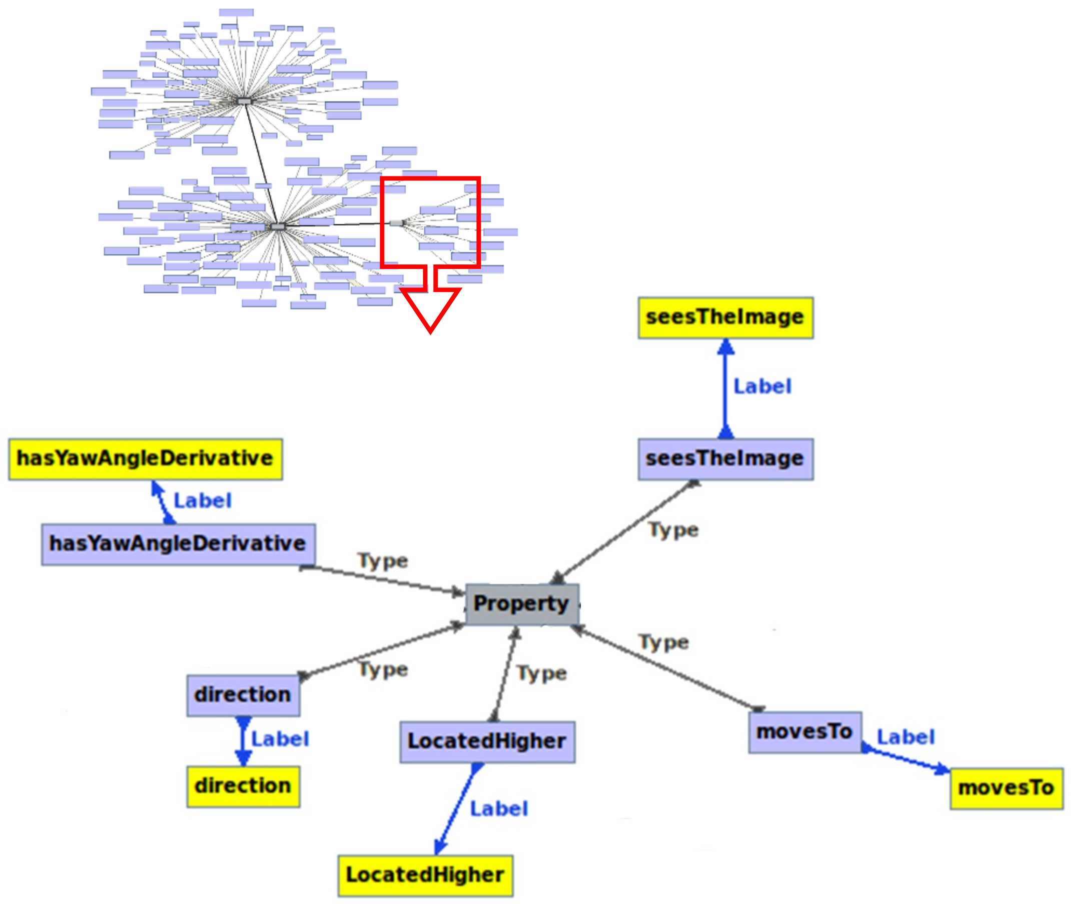


Fig. 5.5: RDF-Graph representing an overview of the RDF-statements stored in the *infobox_property_definition* repository.

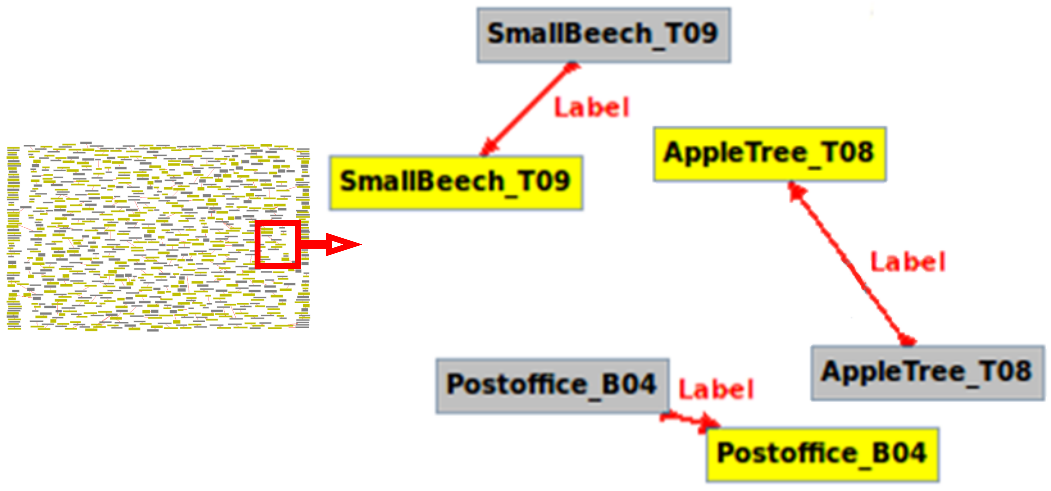


Fig. 5.6: RDF-Graph representing an overview of the RDF-statements stored in *label_en*, each recourse is provided with a label, for this experiment the same resource names is chosen as labels.

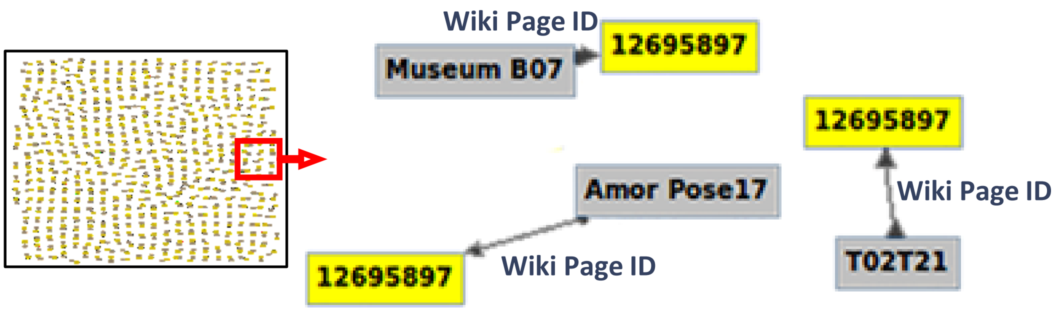


Fig. 5.7: RDF-Graph representing the RDF statements stored in the *page_ids_en repository*, each resource is provided with a Wikipedia page-id, for this experiment the page-id of the University of Siegen has been used.

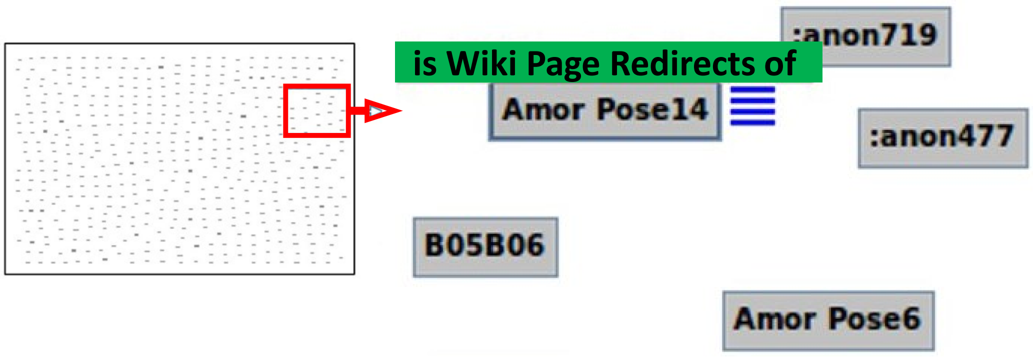
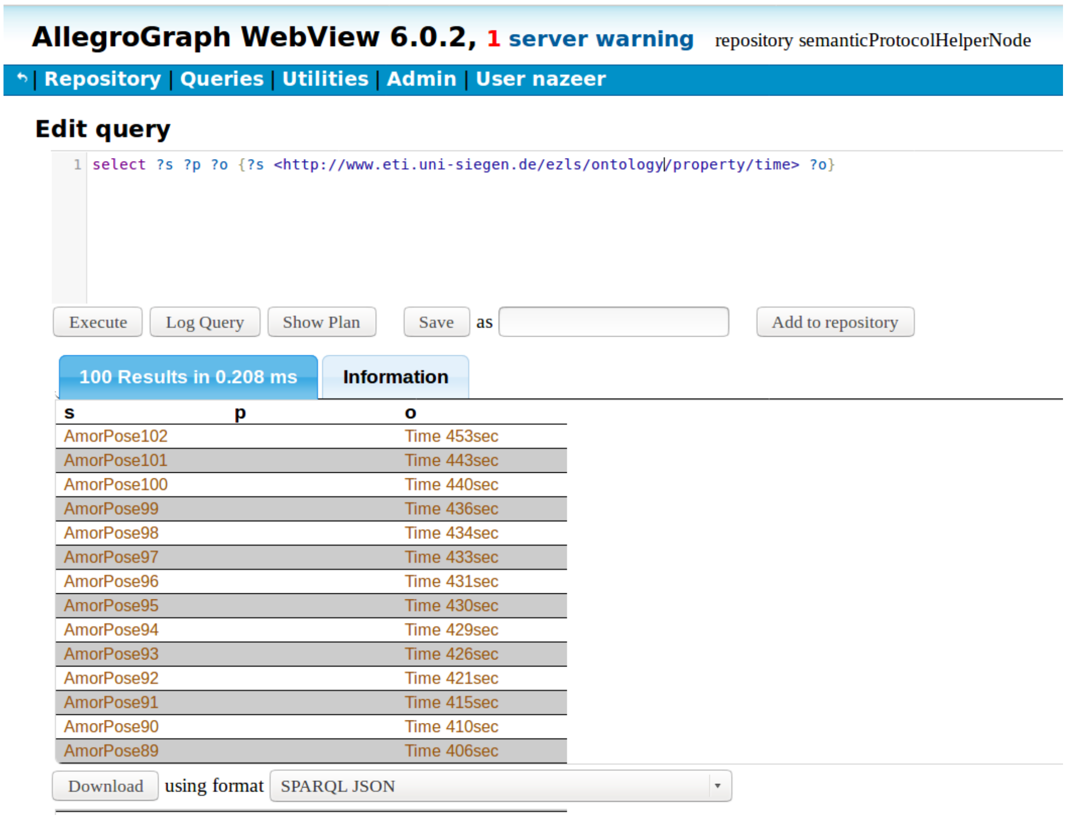


Fig. 5.8: RDF-Graph representing the RDF statements stored in the *redirects_transitive_en repository*, each resource is provided with a "Wiki page redirect" property, this property is useful for redirecting the resources into their associated page-ids.

5.4 Finding specific information

The SPARQL is used to query elements of RDF; however, it is possible to query the semantic information presented in the AMOR core ontology with little or no SPARQL knowledge; this enables non-expert users to query the ontology and search for specific information without expert knowledge of the language. Queries can merely be created with a graph which then itself generates the corresponding SPARQL query. This work utilizes the Gruff which provides the functionality of modeling graphical queries with the "graphical query view" function. An example of querying the AMOR core ontology with the graphical query tool is shown in Fig.5.10. Another advantage of the system is the possibility of remote access and querying data over different platforms. An illustration of the process of querying the semantic information over HTTP in a web-browser by employing the WebView tool of AllegroGraph is shown in Fig.5.9. As can be seen in Fig.5.9 the user can use the web browser as a SPARQL endpoint in which the user can query the semantic information by writing the SPARQL queries in the edit query section shown in Fig.5.9 and the result of the query will be presented accordingly in the result section.

As an alternative of writing SPARQL queries the system provides a graphical query view in which a query can be created by forming a graph, and according to the graph the system generates the proper SPARQL query representing the graph. In the example shown in Fig.5.9 two nodes have been created and linked with the time predicate. After clicking on the run query in the left-hand side of the user interface shown in Fig.5.10a a SPARQL query is generated automatically and also the result of the query is given accordingly, as can be seen in Fig.5.10b.

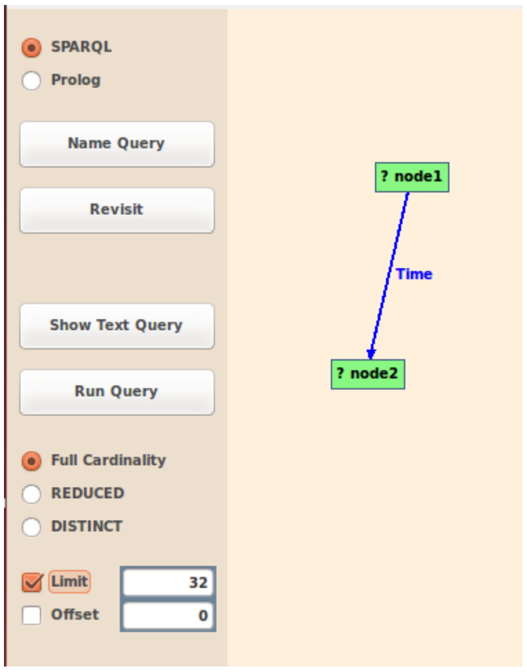

AllegroGraph WebView 6.0.2, 1 server warning
repository semanticProtocolHelperNode
Repository | Queries | Utilities | Admin | User nazeer
Edit query
Execute
Log Query
Show Plan
Save
as
Add to repository
100 Results in 0.208 ms
Information
Download
using format

```
select ?s ?p ?o {?s <http://www.eti.uni-siegen.de/ezls/ontology/property/time> ?o}
```

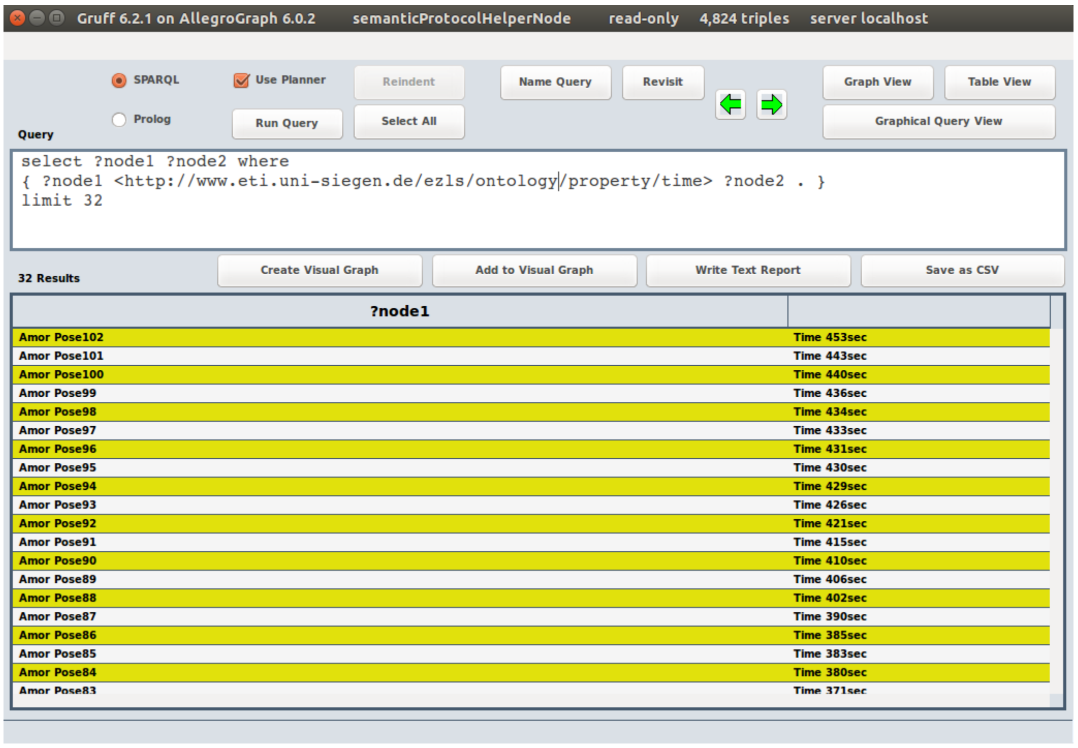
s	p	o
AmorPose102		Time 453sec
AmorPose101		Time 443sec
AmorPose100		Time 440sec
AmorPose99		Time 436sec
AmorPose98		Time 434sec
AmorPose97		Time 433sec
AmorPose96		Time 431sec
AmorPose95		Time 430sec
AmorPose94		Time 429sec
AmorPose93		Time 426sec
AmorPose92		Time 421sec
AmorPose91		Time 415sec
AmorPose90		Time 410sec
AmorPose89		Time 406sec

SPARQL JSON

Fig. 5.9: An example of finding specific information from the collected RDF statements during navigation over HTTP in a web browser by utilizing the WebView.



(a) An example of creating a query by utilizing the graphical query view.



```
select ?node1 ?node2 where
{ ?node1 <http://www.eti.uni-siegen.de/ezls/ontology/property/time> ?node2 . }
limit 32
```

?node1	
Amor Pose102	Time 453sec
Amor Pose101	Time 443sec
Amor Pose100	Time 440sec
Amor Pose99	Time 436sec
Amor Pose98	Time 434sec
Amor Pose97	Time 433sec
Amor Pose96	Time 431sec
Amor Pose95	Time 430sec
Amor Pose94	Time 429sec
Amor Pose93	Time 426sec
Amor Pose92	Time 421sec
Amor Pose91	Time 415sec
Amor Pose90	Time 410sec
Amor Pose89	Time 406sec
Amor Pose88	Time 402sec
Amor Pose87	Time 390sec
Amor Pose86	Time 385sec
Amor Pose85	Time 383sec
Amor Pose84	Time 380sec
Amor Pose83	Time 371sec

(b) The automatic generated SPARQL code and the result of the query created by the graphical query view shown in Fig.5.10a.

Fig. 5.10: An example to demonstrate the use of the graphical query view the gruff for automatic generation of SPARQL queries.

5.4.1 Tracking the movement of AMOR in the environment

During the experimental navigation, AMOR has explored the landscape and the city areas of its environment; this exploring resulted in the creation of 101 AMOR-Pose nodes. These nodes are connected to represent the tour of AMOR in its environment. An overview of the tour of AMOR in the city and landscape proving the time and position of each pose node is shown in Fig.5.11 and Fig.5.12.

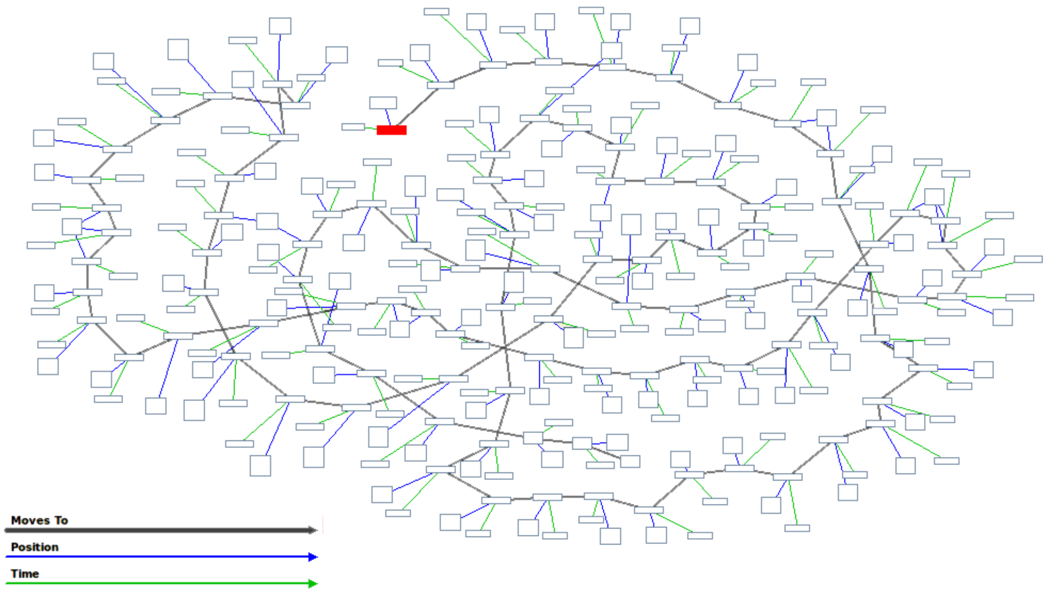


Fig. 5.11: An overview showing the tour of AMOR during the experimental navigation.

5.4.2 The vision of AMOR with its video camera

Along with the semantic camera, AMOR also uses a video camera to capture pixel-images of objects placed in its field of view. Due to the sensitivity of the semantic camera, in some cases, objects detected by the semantic camera might not be visible in the pixel-images captured by the video camera. Hence, and to compensate for the gap between these two cameras, for each pose node three different pixel-images have been captured. Consequently, three predicates have been created to link these images to the pose node in which they have been taken. While exploring, 101 AMOR-Pose nodes have been created which resulted in capturing and recording 303 pixel-images, three images for each pose node. Fig.5.13 is an RDF-Graph that shows pixel images captured by the video camera of AMOR and associated with the AMOR-Pose 101.

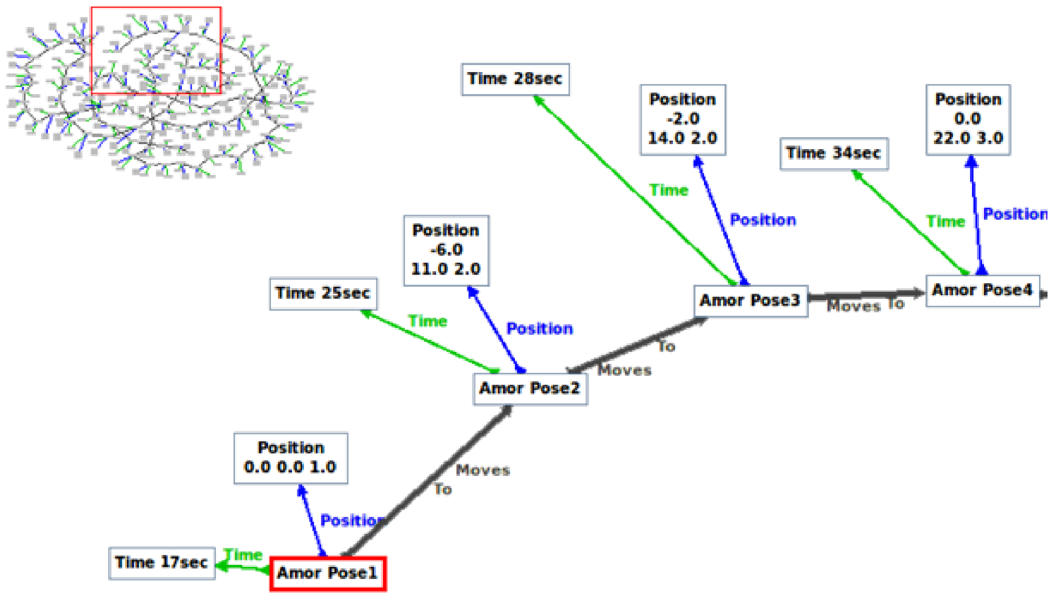


Fig. 5.12: An RDF-Graph showing the subset of the tour of AMOR, nodes has been placed randomly in the graph.

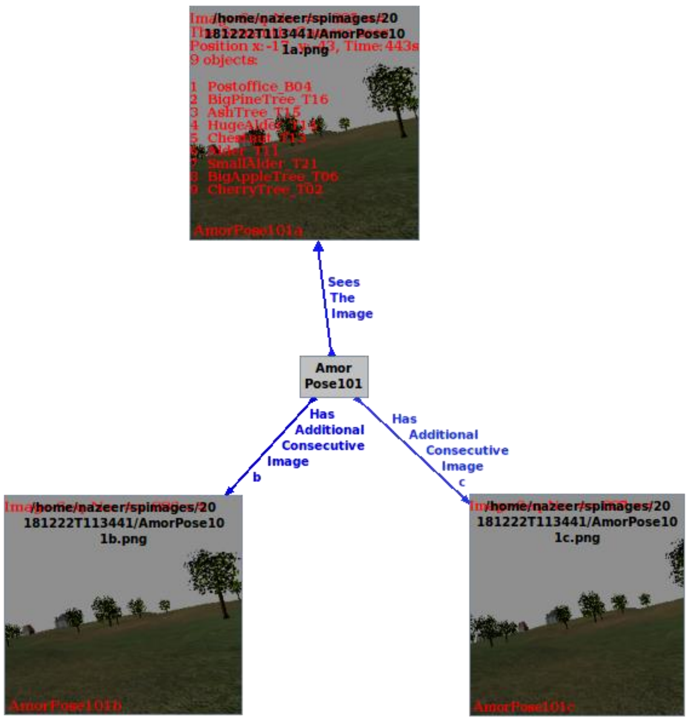

/home/nazeer/spimages/20181222T113441/AmorPose101a.png
Position x: -17, y: 43, Time: 443s
9 objects:
1 Postoffice_B04
2 BigPineTree_T16
3 AshTree_T15
4 HugeAlder_T14
5 Chestnut_T13
6 Alder_T11
7 SmallAlder_T21
8 BigAppleTree_T06
9 CherryTree_T02
AmorPose101a
Sees The Image
Amor Pose101
Has Additional Consecutive Image b
Has Additional Consecutive Image c
/home/nazeer/spimages/20181222T113441/AmorPose101b.png
AmorPose101b
/home/nazeer/spimages/20181222T113441/AmorPose101c.png
AmorPose101c

Fig. 5.13: An RDF-Graph representing the vision of AMOR with its video camera.

5.4.3 Finding a specific node

The RoboSemProc provides the ability to extract all information about a specific node; it can either be a pose node or a specific landmark. To illustrate, the following example, in Fig.5.14, shows all information about AMOR-Pose 86.

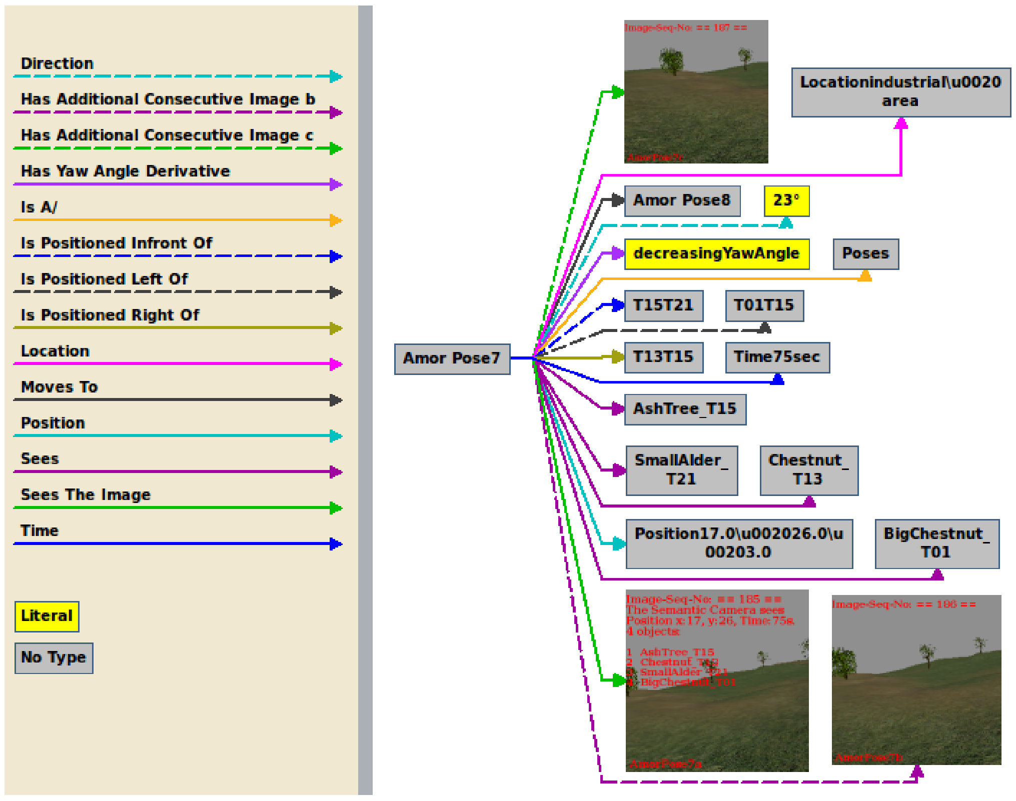


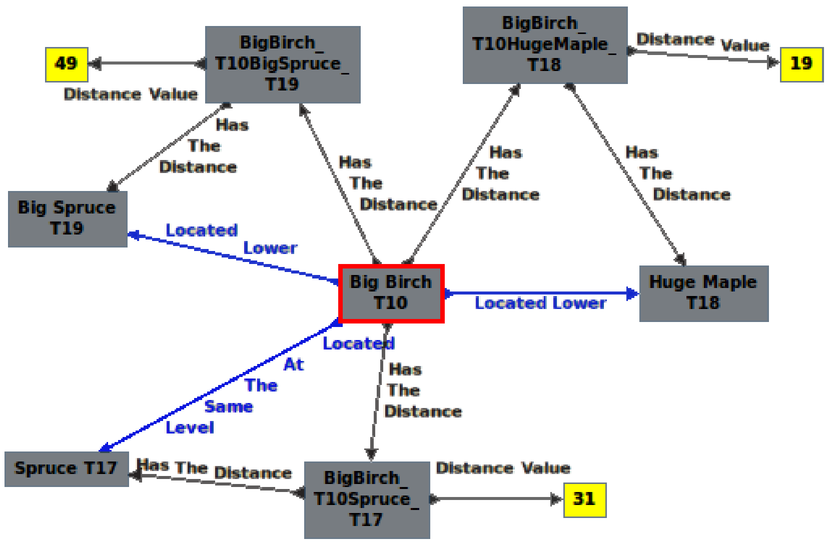
Fig. 5.14: An RDF-Graph representing all information about AMOR-Pose 7.

In the same way, the information about any other node can be extracted and shown in RDF-graphs.

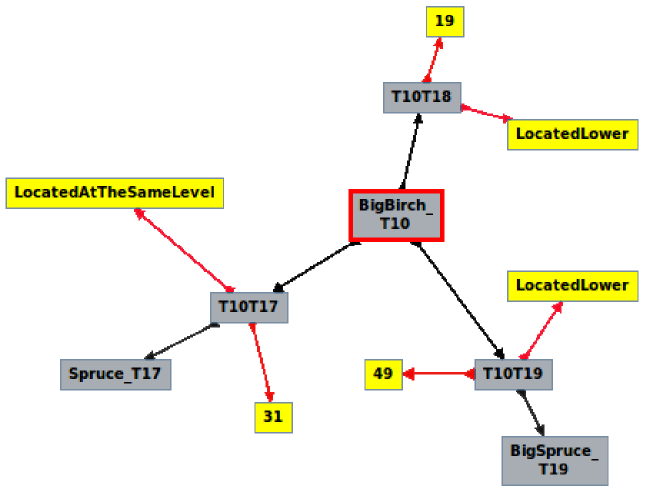
5.4.4 Environmental perception of and inter-object relations

One of the significant aspects of the AMOR core ontology is its ability to express inter-object relations and particularly relations expressed by using the spatial preposition. It compensates for the gap between human and robot by enabling AMOR to describe the inter-object relations between related landmarks in the same way that human describes them in his/her daily life. To model inter-object relations, different approaches have been implemented, and a new approach has been proposed. During exploration, the AMOR core ontology has been populated with a set of relations representing the inter-object relations between related landmarks. In the following, the result of each approach for modeling these relations is described.

- The distance and height relations: While exploring its environment, in a number of poses AMOR has detected more than one landmark. In these poses, the position information of a detected landmark, which is provided by the semantic camera, has been used to model RDF-statements representing the distance and height difference between detected landmarks. These statements have been modeled with different approaches and have instantiated the AMOR core ontology. Fig.5.15 shows the RDF-Graphs representing the distance and height relations between BigBreech_T10 and its neighbors created, during the experimental navigation, in both blank-node and helperNode approaches.



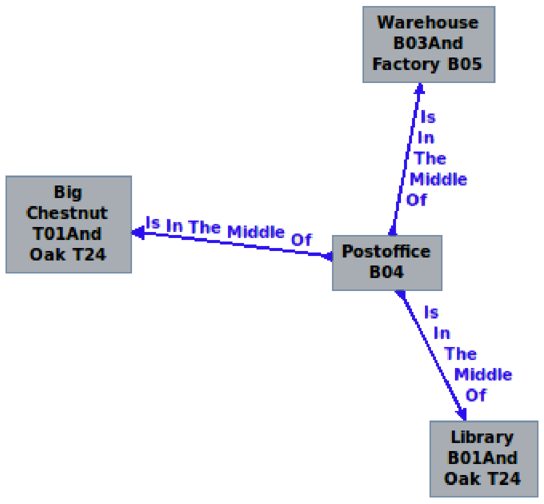
(a) An RDF-Graph illustrating the distance and height difference between BigBirch_T10 and its neighbors in the blank-node approach.



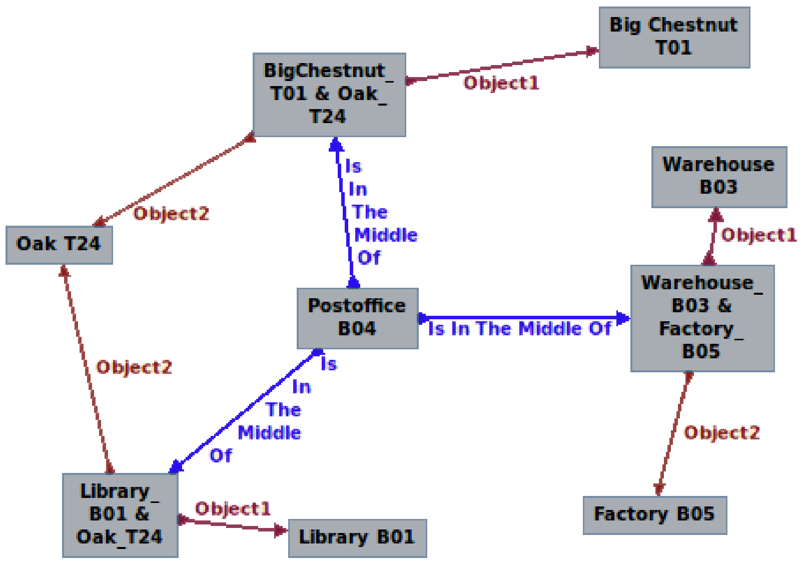
(b) An RDF-Graph illustrating the distance and height difference between BigBirch_T10 and its neighbors in the helperNode approach.

Fig. 5.15: An illustration of the distance and height difference between BigBirch_T10 and its neighbors in both blank-node and helperNode approaches.

- The middle of relation: While exploring the city, AMOR has detected a number of landmarks. Among them, the Postoffice_B04 appeared to be located in the area close to the middle point of the Warehouse_B03 and Factory_B05, BigChestnut_T01 and Oak_T24, and also Library_B01 and Oak_T24. This information has been collected and represented in RDF format and stored in the AMOR core ontology. RDF-statements representing this information have been modeled in different approaches, to illustrate, Fig.5.16 and 5.17 show this information in created while navigation in real-time in different approaches and stored in the AMOR core ontology.



(a) RDF-Graph representing the middle of relation created in the binary approach.



(b) RDF-Graph representing the middle of relation created in the blank-node approach.

Fig. 5.16: An illustration of the middle of relation, from information created in real-time during navigation.

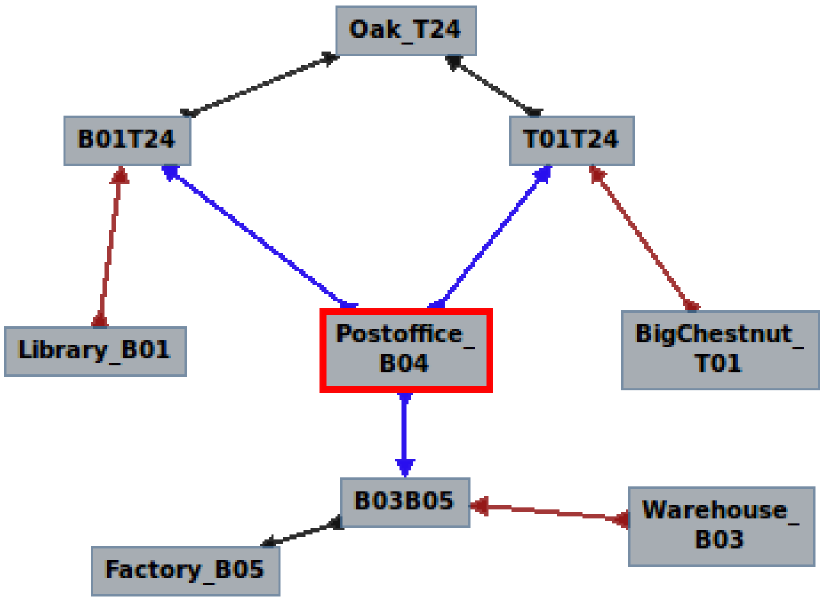
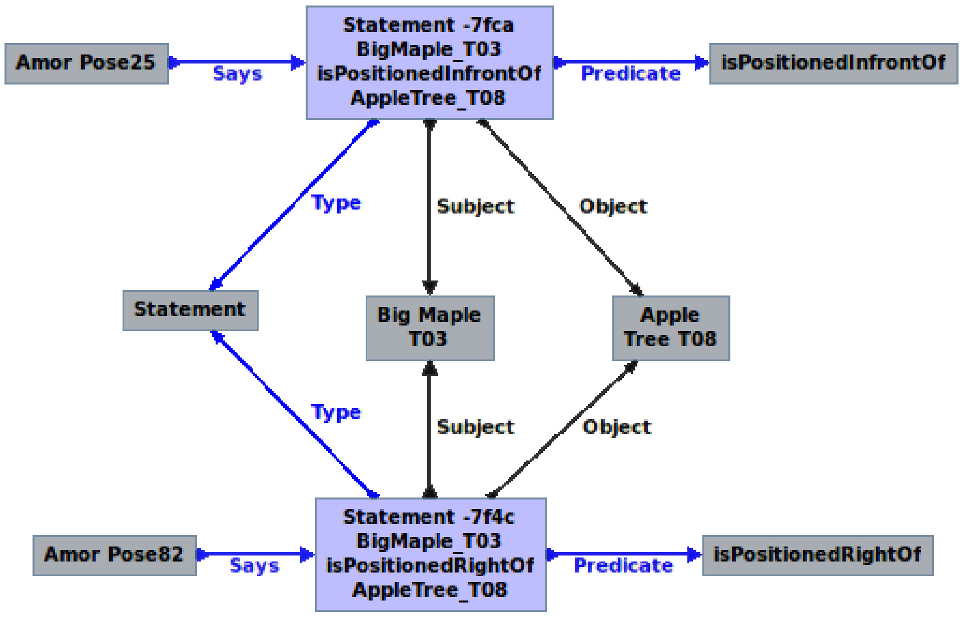


Fig. 5.17: RDF-Graph representing the middle of relation created in the helperNode approach. The blue arrow represent in middle of relation, red represents *towardHelperNode* and black represents *fromHelperNode*.

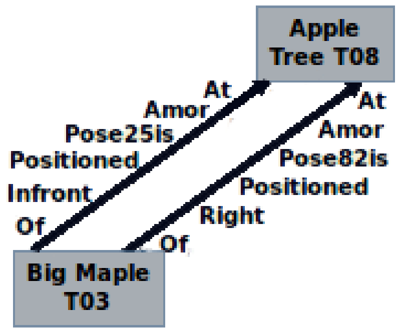
The subsequent code section demonstrates the use of SPARQL for querying the middle of relation in the binary, blank-node and helperNode approaches.

```
----blank-node approach
select ?s ?o ?p where
{
 ?object .
<http://www.eti.uni-siegen.de/ezls/ontology/resource/Postoffice\_B04>
<http://www.eti.uni-siegen.de/ezls/ontology/property/isInTheMiddleOf>
 ?object .
    }

----helperNode approach
select ?subject ?object ?object2 where
{
<http://www.eti.uni-siegen.de/ezls/ontology/resource/Postoffice\_B04>
<http://www.eti.uni-siegen.de/ezls/ontology/property/isInTheMiddleOf>
 ?subject .

 ?object
<http://www.eti.uni-siegen.de/ezls/ontology/property/towardHelperNode>
 ?subject .
 ?subject
<http://www.eti.uni-siegen.de/ezls/ontology/property/fromHelperNode>
 ?subject2 .
  }

----binary approach
select ?subject where
{
<http://www.eti.uni-siegen.de/ezls/ontology/resource/Postoffice\_B04>
<http://www.eti.uni-siegen.de/ezls/ontology/property/isInTheMiddleOf>
 ?subject .
    }
```

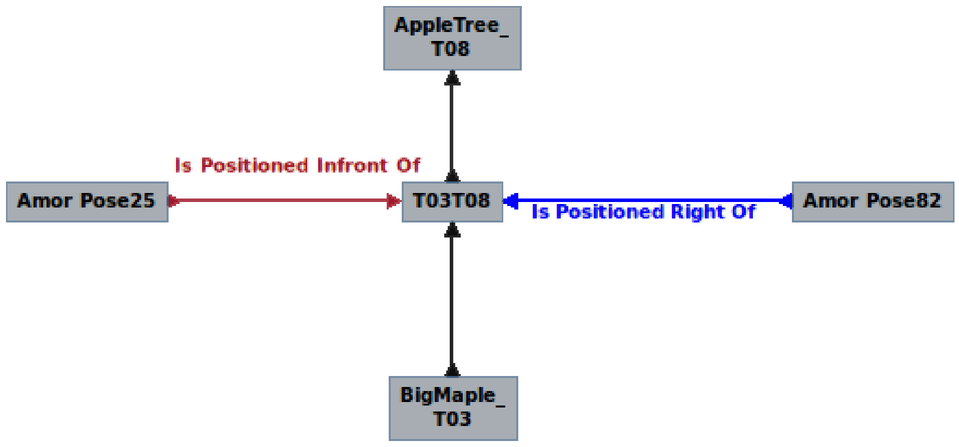
- Spatial relations: After navigating for a while in the landscape, at pose 25, AMOR approaches the BigMaple_T03 and the AppleTree_T08. At this pose, the point of view of AMOR indicates that the BigMaple_T03 is located in front of the AppleTree_T08. This information is recorded and used to create RDF-statements representing the relation of the BigMaple_T03 to the AppleTree_T08 looking from a specific position and orientation. AMOR continues its tour and at pose 82 again detects the BigMaple_T03 and AppleTree_T08; this time looking from a different position AMOR indicates that the BigMaple_T03 is located on the right of the AppleTree_T08; This information is also used to create RDF-statements representing the location of two landmarks from a different point of view. Fig.5.18 demonstrates that the AMOR core ontology has been instantiated with RDF statements representing the mentioned scenario.



(a) An RDF-Graph representing inter-object relations created in the blank-node approach.


Apple Tree T08
At Amor Pose25is Positioned Infront Of
At Amor Pose82is Positioned Right Of
Big Maple T03

(b) An RDF-Graph representing inter-object relations created in the binary approach.



(c) An RDF-Graph representing inter-object relations created in the helperNode approach.

Fig. 5.18: An illustration of inter-object relations, from information captured during navigation.

For modeling relations between landmarks, various approaches have been implemented. The proposed approach is, however, superior in a way that it enables the RoboSemProc to represent the same information with a fewer number of predicates. Another advantage of the proposed approach is its ability to represent all relations between related landmarks in a single helperNode. In this way, a single SPARQL query is efficient to query all relations between related landmarks. The subsequent SPARQL code is an example to illustrate the use of a single helperNode in representing all relations between the Museum_B07 and Church_B09. The RDF-Graph outcome of the SPARQL query is shown in Fig.5.19.

```
select ?s ?o ?p1 ?p2 where
{ ?o ?p1 <http://www.eti.uni-siegen.de/ezls/ontology/resource/B07B09> .
  <http://www.eti.uni-siegen.de/ezls/ontology/resource/B07B09> ?p2 ?s .
  }
```

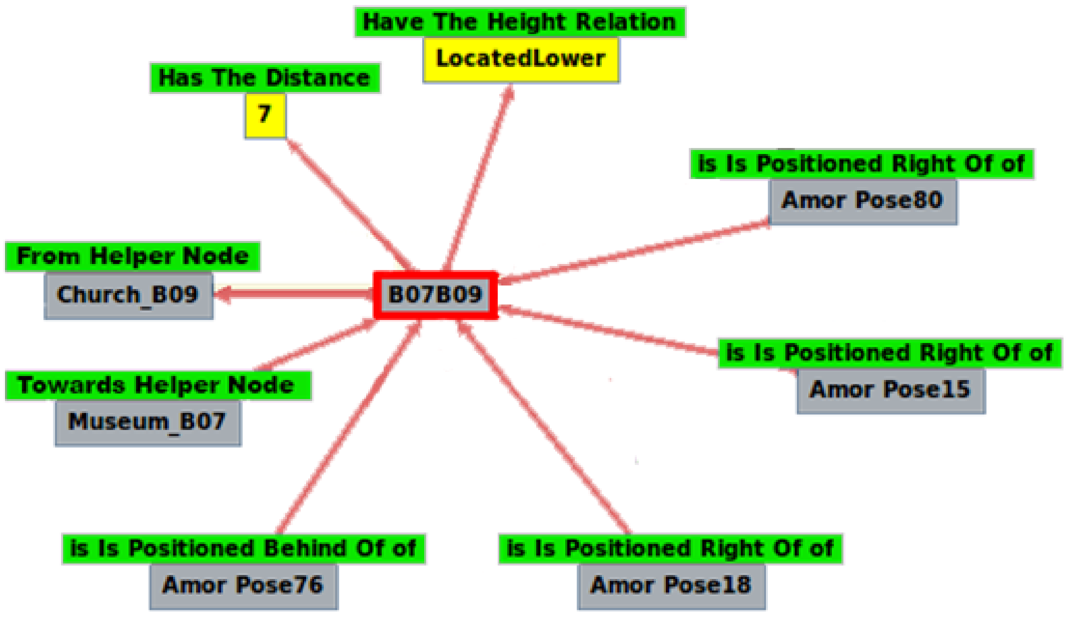


Fig. 5.19: An RDF-Graph representing the application of the helperNode in expressing all relations between two related landmarks.

5.4.5 Finding landmarks by their type

While AMOR was exploring the city and the landscape, its semantic camera provided information about the landmarks placed on the campus and in industrial areas as well. Each detected landmark is annotated with its type which has been used to associate them to their corresponding classes in the AMOR core ontology. According to their types, tree type landmarks are associated with the tree class, and the building type landmarks are associated with the building class. As a result, it is possible to query landmarks by their type. To illustrate, Fig.5.20, and Fig.5.21 show the detected building by AMOR and their class hierarchy in the AMOR core ontology.

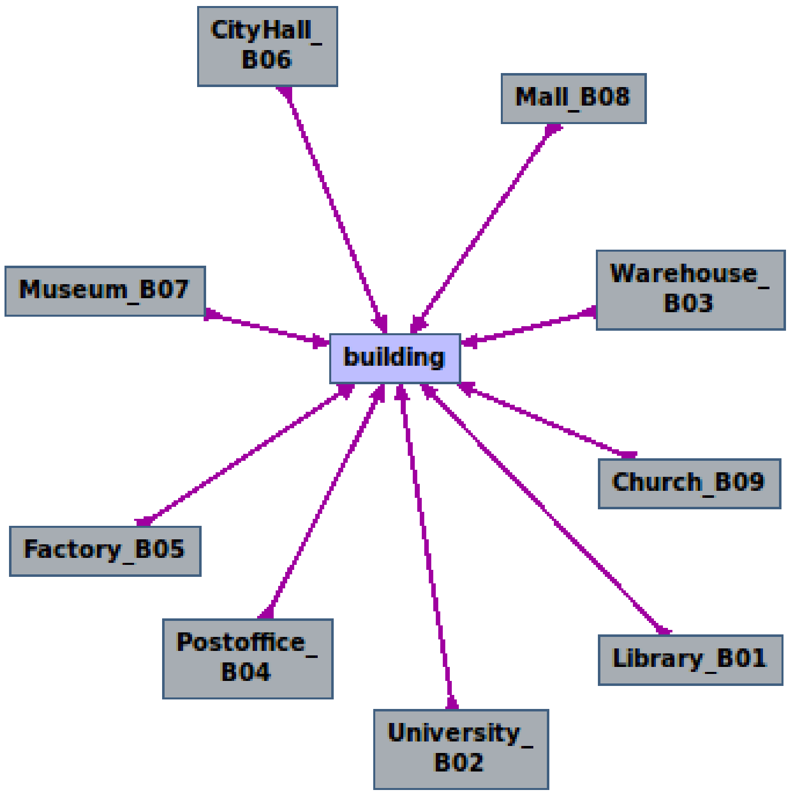


Fig. 5.20: An RDF-Graph representing all detected buildings by AMOR, during its exloring in the city and the landscape.

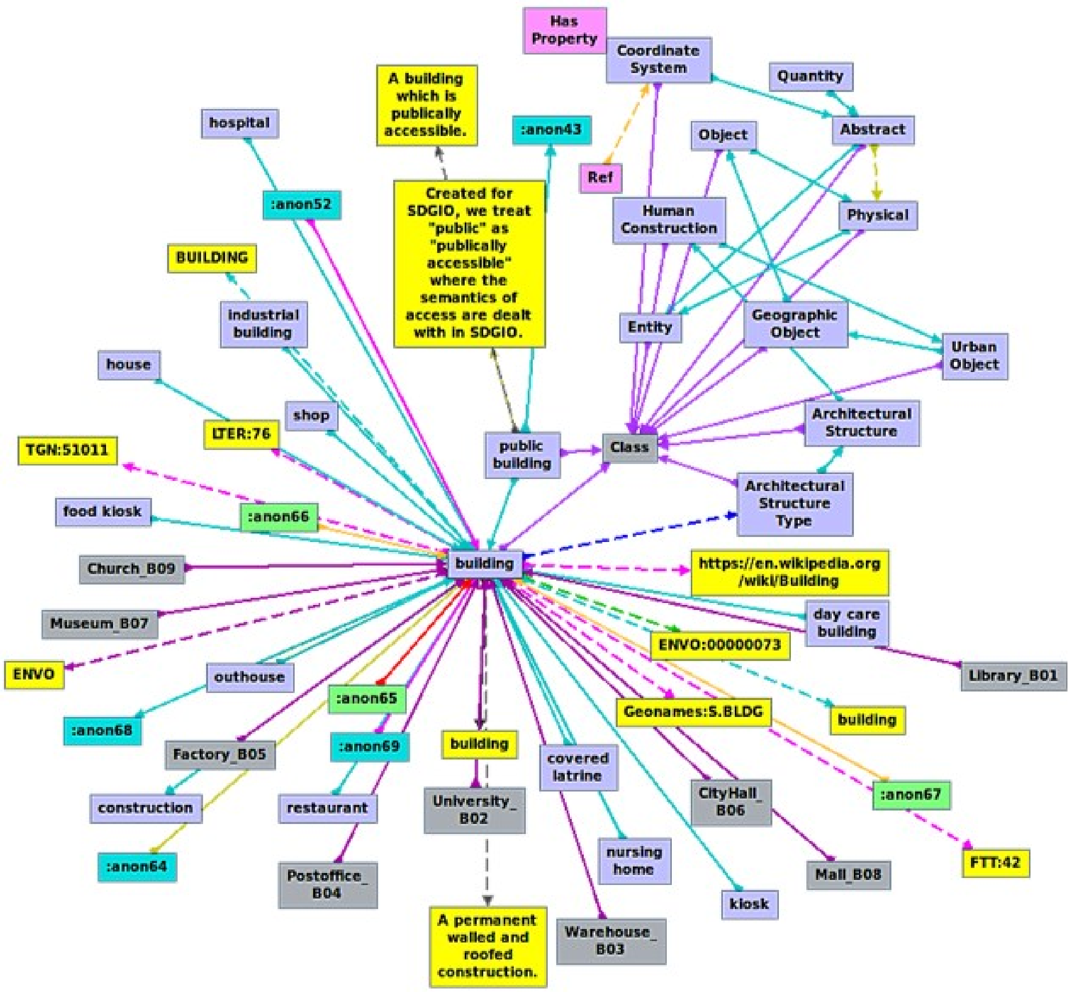


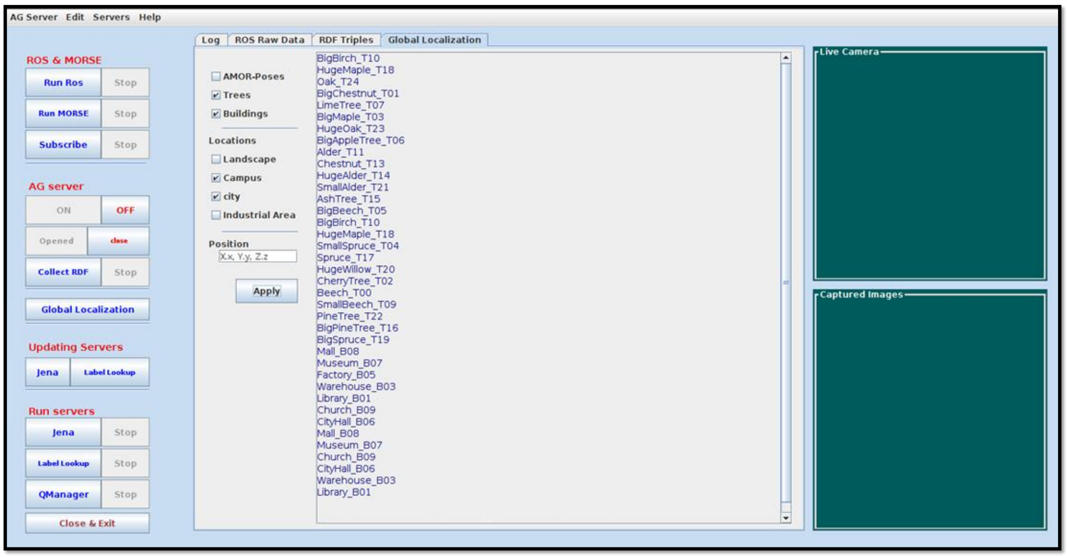
Fig. 5.21: An RDF-Graph representing all detected buildings by AMOR, during its exloring in the city and the landscape with a different layout.

5.5 Global localization

As can be seen in the user interface of the RoboSemProc, the system provides the simple search functionality to search for specific AMOR-Poses or landmarks based on their type, position or location. The global localization button will direct the user to the global localization tab in which users can use simple check boxes to search for specific information. For instance, as shown in Fig.5.22, it is possible to output all landmarks which are located in the campus and the city.

5.6 Natural language communication

During the experimental navigation, AMOR populated the AMOR core ontology with new statements representing the tour of AMOR and its vision of the environment.



☐ AMOR-Poses
☑ Trees
☑ Buildings

Locations
☐ Landscape
☑ Campus
☑ city
☐ Industrial Area

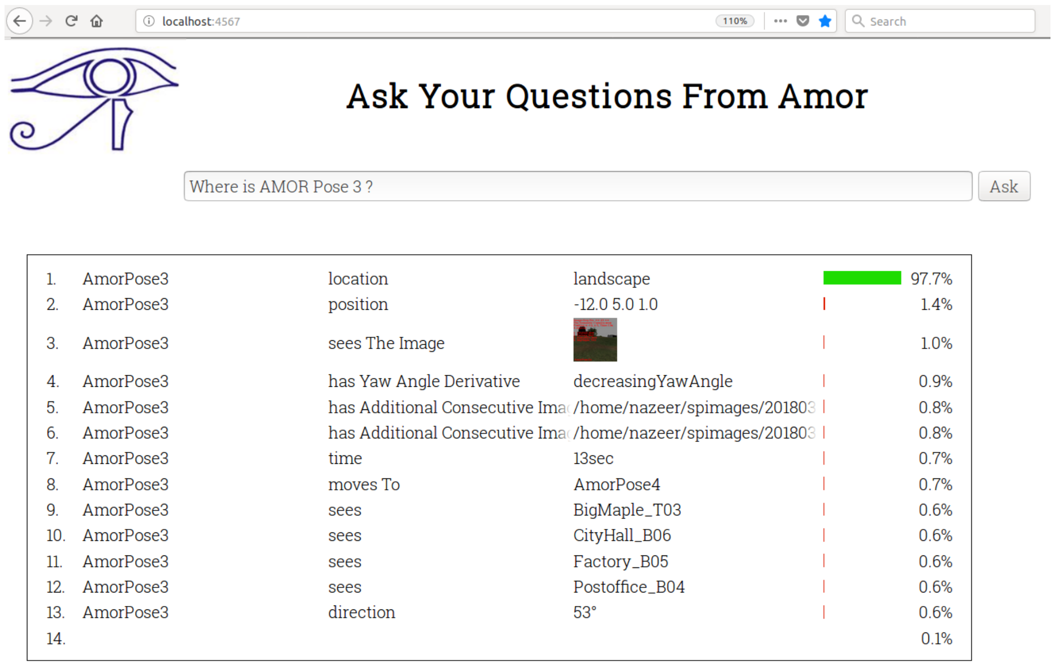
Position: X.x, Y.y, Z.z

```
BigBirch_T10
HugeMaple_T18
Oak_T24
BigChestnut_T01
LimeTree_T07
BigMaple_T03
HugeOak_T23
BigAppleTree_T06
Alder_T11
Chestnut_T13
HugeAlder_T14
SmallAlder_T21
AshTree_T15
BigBeech_T05
BigBirch_T10
HugeMaple_T18
SmallSpruce_T04
Spruce_T17
HugeWillow_T20
CherryTree_T02
Beech_T00
SmallBeech_T09
PineTree_T22
BigPineTree_T16
BigSpruce_T19
Mall_B08
Museum_B07
Factory_B05
Warehouse_B03
Library_B01
Church_B09
CityHall_B06
Mall_B08
Museum_B07
Church_B09
CityHall_B06
Warehouse_B03
Library_B01
```

Fig. 5.22: An overview of the global localization functionality of the RoboSemProc.

This process can be repeated multiple of times. New navigation can be conducted on top of last navigations, and the AMOR core ontology can be instantiated and enriched with new information each time a tour is conducted. The information collected from different runs are collected and merged into one consistent ontology. In order to be able to communicate with AMOR about its last tour, it is essential to update the YodaQA system with the information collected from the exploration. The RoboSemProc provides the functionality of updating the Jena fuseki server and the label-lookup server on request. Therefore, it is possible to update the servers used for the natural language communication after each navigation. As can be seen in Fig.5.1 the user interface of the RoboSemProcs provides a set of buttons that are used for updating and running the YodaQA system.

After updating servers, and the successful running of both the Jena and label-lookup server, the QManager button will then run the natural language communication system and redirect the user to a web browser in which the communication with AMOR is possible. The system enables users to ask questions regarding AMOR poses and landmarks in a natural language. Questions can be about the tour of AMOR like its position, location, time, direction, yaw angle derivative, or its semantic such as the position, location, type, vicinity type, or description of detected objects and also graphic vision such as the images taken at each pose. The QA system is, however, inadequate to answer over RDF n-ary relations regarding the inter-object relations between landmarks of environments, to overcome this limitation the describe landmark can be used to output all inter-object relations between the landmark and its related landmarks in the environment. In the following, some examples are given.

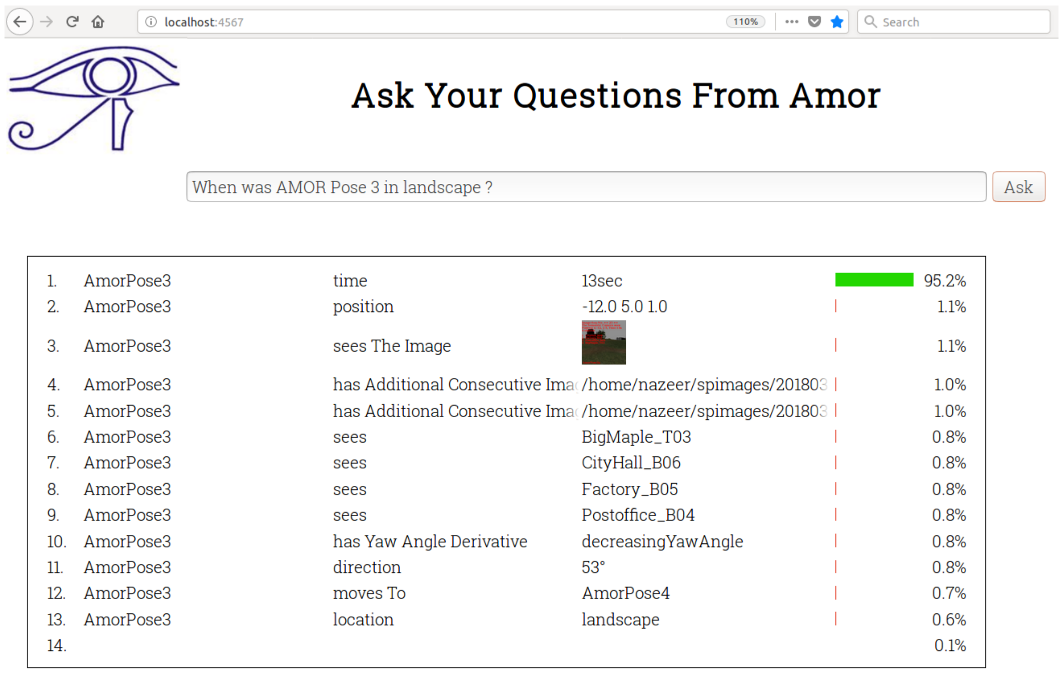


Ask Your Questions From Amor

Where is AMOR Pose 3 ?

1.	AmorPose3	location	landscape	97.7%
2.	AmorPose3	position	-12.0 5.0 1.0	1.4%
3.	AmorPose3	sees The Image		1.0%
4.	AmorPose3	has Yaw Angle Derivative	decreasingYawAngle	0.9%
5.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	0.8%
6.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	0.8%
7.	AmorPose3	time	13sec	0.7%
8.	AmorPose3	moves To	AmorPose4	0.7%
9.	AmorPose3	sees	BigMaple_T03	0.6%
10.	AmorPose3	sees	CityHall_B06	0.6%
11.	AmorPose3	sees	Factory_B05	0.6%
12.	AmorPose3	sees	Postoffice_B04	0.6%
13.	AmorPose3	direction	53°	0.6%
14.				0.1%

Fig. 5.23: A question regarding the location of AMOR-Pose3.

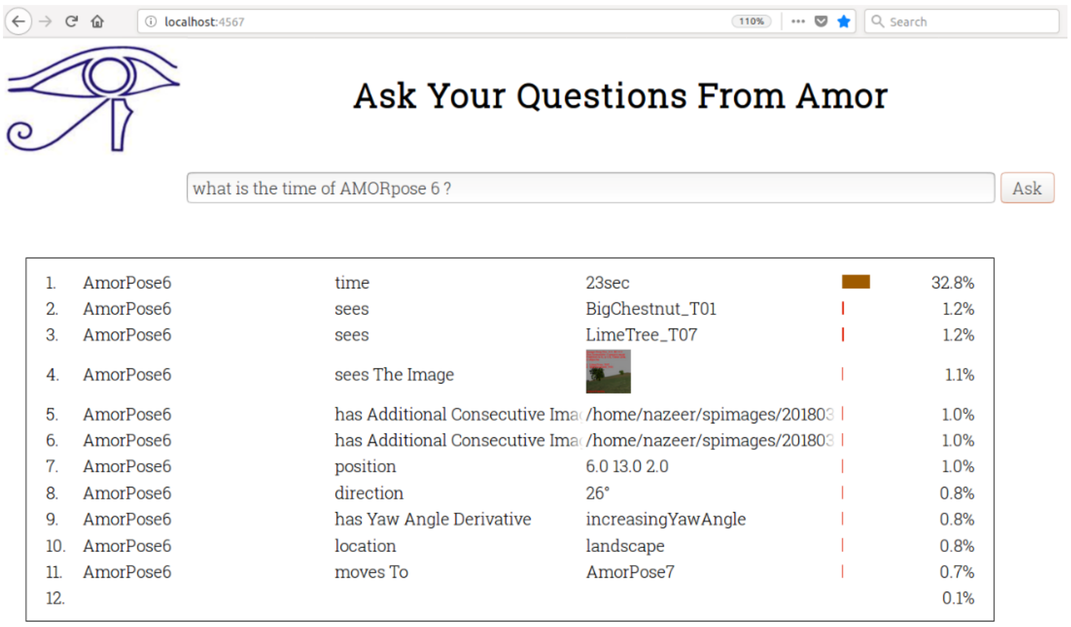


Ask Your Questions From Amor

When was AMOR Pose 3 in landscape ?

1.	AmorPose3	time	13sec	95.2%
2.	AmorPose3	position	-12.0 5.0 1.0	1.1%
3.	AmorPose3	sees The Image		1.1%
4.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.0%
5.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.0%
6.	AmorPose3	sees	BigMaple_T03	0.8%
7.	AmorPose3	sees	CityHall_B06	0.8%
8.	AmorPose3	sees	Factory_B05	0.8%
9.	AmorPose3	sees	Postoffice_B04	0.8%
10.	AmorPose3	has Yaw Angle Derivative	decreasingYawAngle	0.8%
11.	AmorPose3	direction	53°	0.8%
12.	AmorPose3	moves To	AmorPose4	0.7%
13.	AmorPose3	location	landscape	0.6%
14.				0.1%

Fig. 5.24: A question regarding the time of AMOR-Pose3, asking "when was the time", the system returns the correct answer, with the rank of 95.2%.


localhost:4567

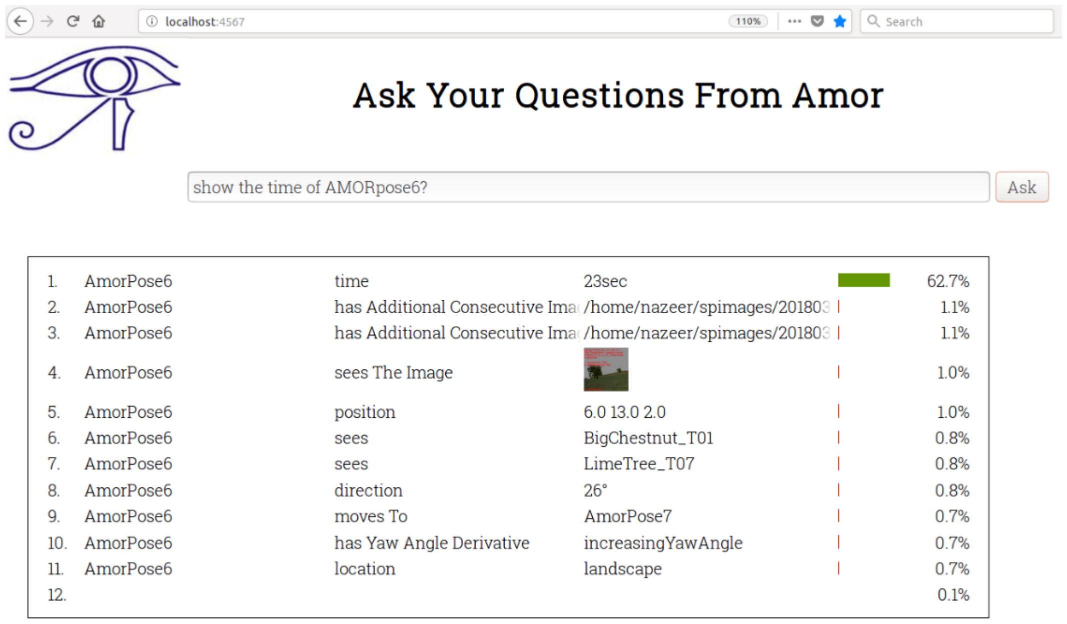
Ask Your Questions From Amor

what is the time of AMORpose 6 ?

Ask

1.	AmorPose6	time	23sec	32.8%
2.	AmorPose6	sees	BigChestnut_T01	1.2%
3.	AmorPose6	sees	LimeTree_T07	1.2%
4.	AmorPose6	sees The Image		1.1%
5.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.0%
6.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.0%
7.	AmorPose6	position	6.0 13.0 2.0	1.0%
8.	AmorPose6	direction	26°	0.8%
9.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	0.8%
10.	AmorPose6	location	landscape	0.8%
11.	AmorPose6	moves To	AmorPose7	0.7%
12.				0.1%

Fig. 5.25: A question regarding the time of AMOR-Pose6, asking "what was the time", the system returns the correct answer, with the rank of 32.8%.


localhost:4567

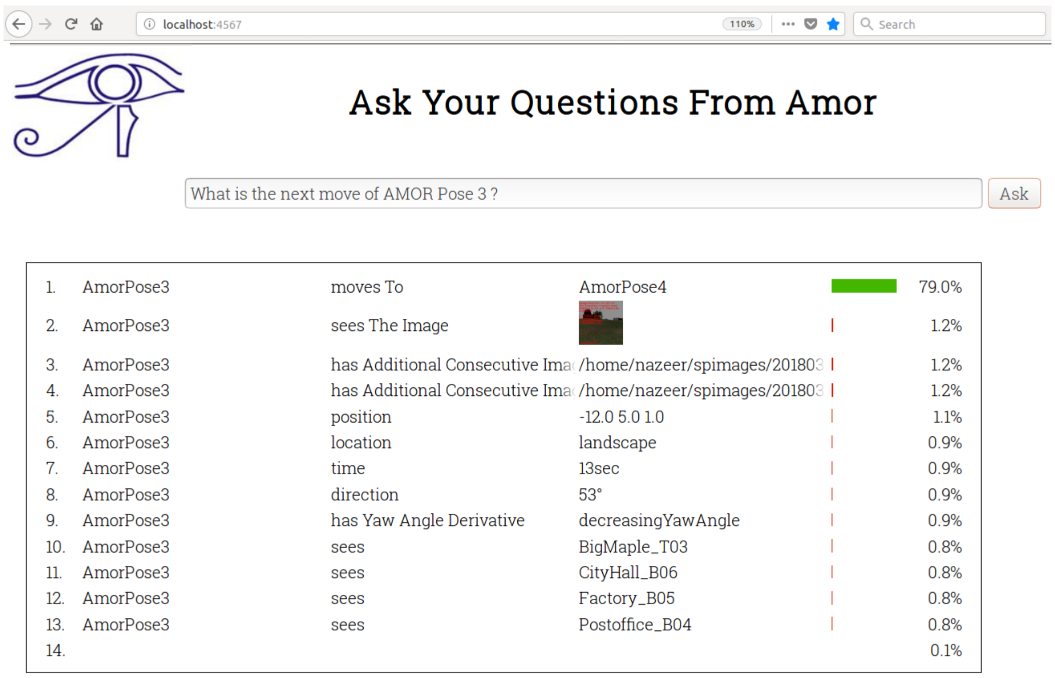
Ask Your Questions From Amor

show the time of AMORpose6?

Ask

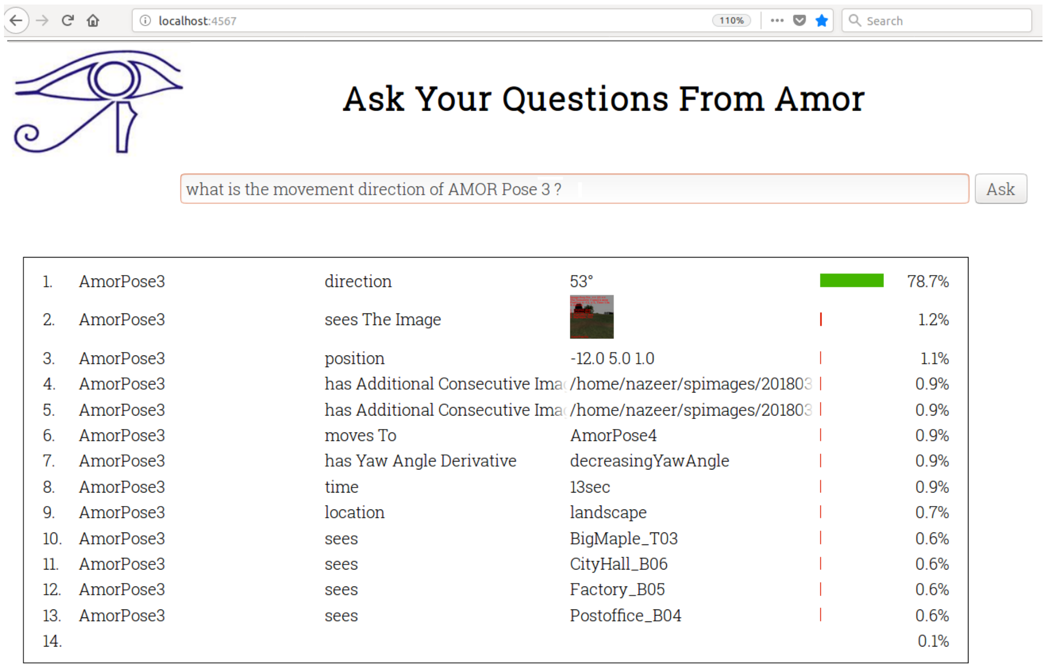
1.	AmorPose6	time	23sec	62.7%
2.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.1%
3.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.1%
4.	AmorPose6	sees The Image		1.0%
5.	AmorPose6	position	6.0 13.0 2.0	1.0%
6.	AmorPose6	sees	BigChestnut_T01	0.8%
7.	AmorPose6	sees	LimeTree_T07	0.8%
8.	AmorPose6	direction	26°	0.8%
9.	AmorPose6	moves To	AmorPose7	0.7%
10.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	0.7%
11.	AmorPose6	location	landscape	0.7%
12.				0.1%

Fig. 5.26: A question regarding the time of AMOR-Pose6, asking "show the time", the system returns the correct answer, with the rank of 62.7%.


localhost:4567
110%
Search
Ask Your Questions From Amor
What is the next move of AMOR Pose 3 ?
Ask

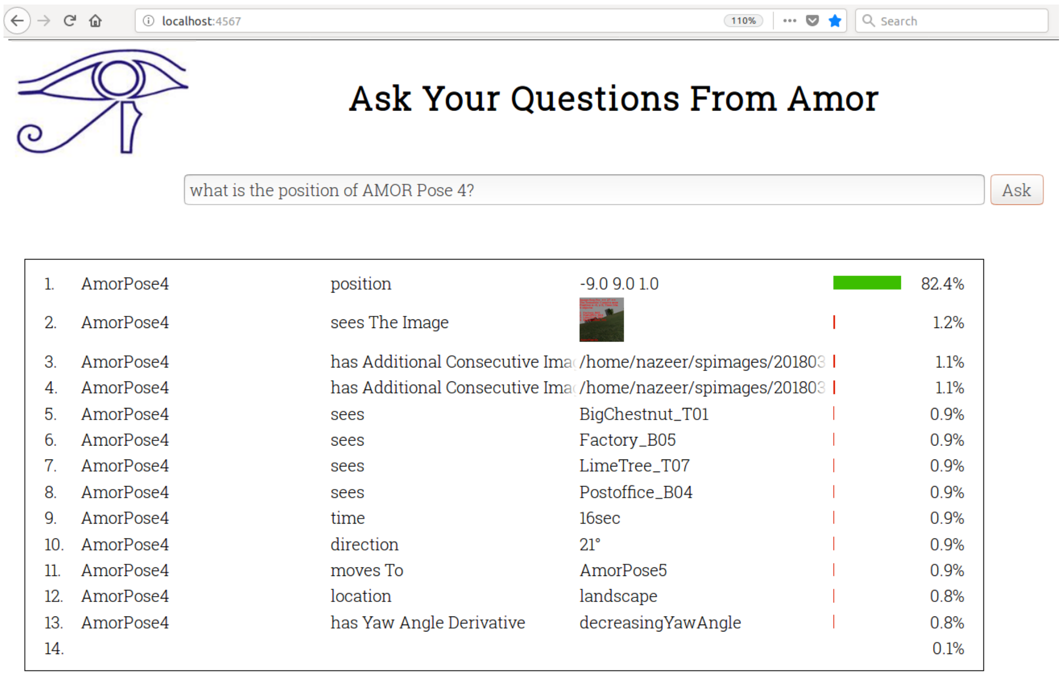
1.	AmorPose3	moves To	AmorPose4	79.0%
2.	AmorPose3	sees The Image		1.2%
3.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.2%
4.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.2%
5.	AmorPose3	position	-12.0 5.0 1.0	1.1%
6.	AmorPose3	location	landscape	0.9%
7.	AmorPose3	time	13sec	0.9%
8.	AmorPose3	direction	53°	0.9%
9.	AmorPose3	has Yaw Angle Derivative	decreasingYawAngle	0.9%
10.	AmorPose3	sees	BigMaple_T03	0.8%
11.	AmorPose3	sees	CityHall_B06	0.8%
12.	AmorPose3	sees	Factory_B05	0.8%
13.	AmorPose3	sees	Postoffice_B04	0.8%
14.				0.1%

Fig. 5.27: A question regarding the next AMOR-Pose which follows AMOR-Pose3.


localhost:4567
110%
Search
Ask Your Questions From Amor
what is the movement direction of AMOR Pose 3 ?
Ask

1.	AmorPose3	direction	53°	78.7%
2.	AmorPose3	sees The Image		1.2%
3.	AmorPose3	position	-12.0 5.0 1.0	1.1%
4.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	0.9%
5.	AmorPose3	has Additional Consecutive Ima	/home/nazeer/spimages/20180	0.9%
6.	AmorPose3	moves To	AmorPose4	0.9%
7.	AmorPose3	has Yaw Angle Derivative	decreasingYawAngle	0.9%
8.	AmorPose3	time	13sec	0.9%
9.	AmorPose3	location	landscape	0.7%
10.	AmorPose3	sees	BigMaple_T03	0.6%
11.	AmorPose3	sees	CityHall_B06	0.6%
12.	AmorPose3	sees	Factory_B05	0.6%
13.	AmorPose3	sees	Postoffice_B04	0.6%
14.				0.1%

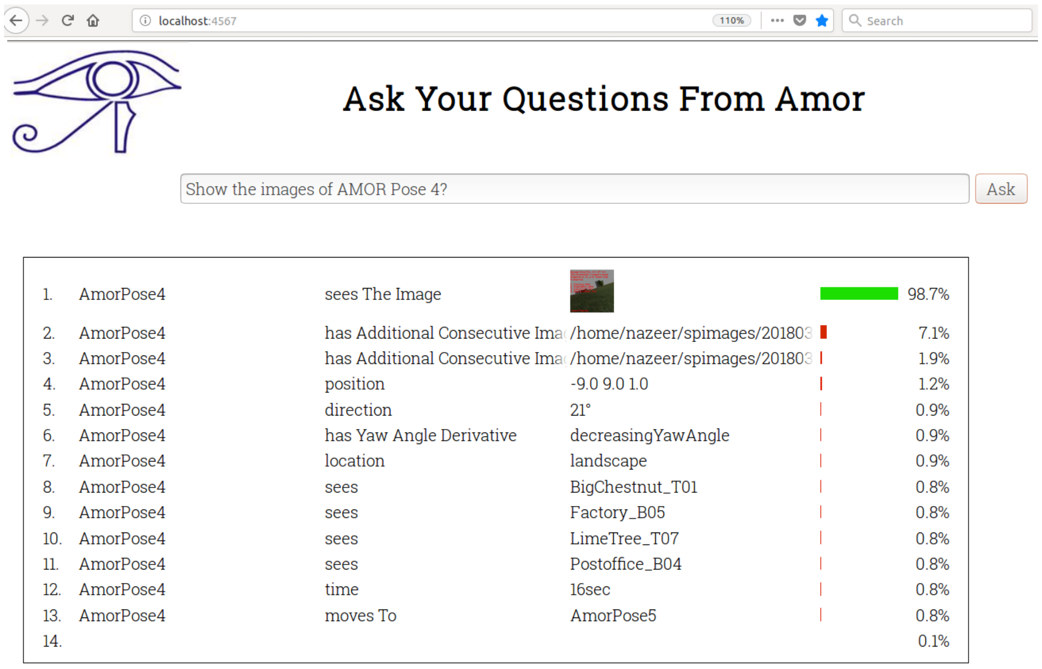
Fig. 5.28: A question regarding the vector direction between AMOR-Pose3 and 4.


Ask Your Questions From Amor

what is the position of AMOR Pose 4?

1.	AmorPose4	position	-9.0 9.0 1.0	82.4%
2.	AmorPose4	sees The Image		1.2%
3.	AmorPose4	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.1%
4.	AmorPose4	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.1%
5.	AmorPose4	sees	BigChestnut_T01	0.9%
6.	AmorPose4	sees	Factory_B05	0.9%
7.	AmorPose4	sees	LimeTree_T07	0.9%
8.	AmorPose4	sees	Postoffice_B04	0.9%
9.	AmorPose4	time	16sec	0.9%
10.	AmorPose4	direction	21°	0.9%
11.	AmorPose4	moves To	AmorPose5	0.9%
12.	AmorPose4	location	landscape	0.8%
13.	AmorPose4	has Yaw Angle Derivative	decreasingYawAngle	0.8%
14.				0.1%

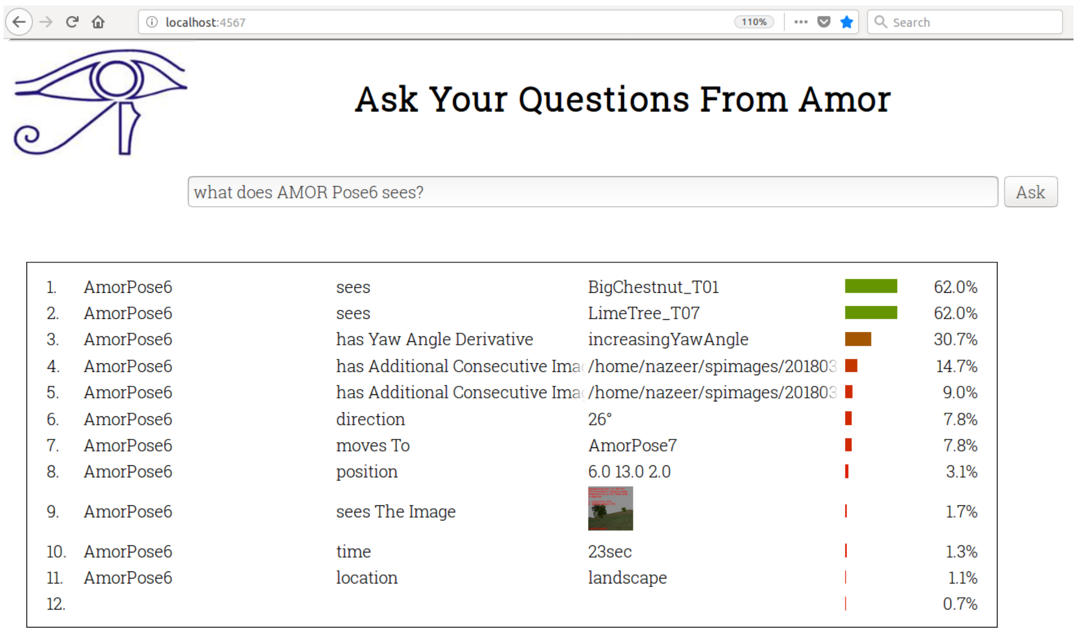
Fig. 5.29: A question regarding the position of AMOR-Pose4.


Ask Your Questions From Amor

Show the images of AMOR Pose 4?

1.	AmorPose4	sees The Image		98.7%
2.	AmorPose4	has Additional Consecutive Ima	/home/nazeer/spimages/20180	7.1%
3.	AmorPose4	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.9%
4.	AmorPose4	position	-9.0 9.0 1.0	1.2%
5.	AmorPose4	direction	21°	0.9%
6.	AmorPose4	has Yaw Angle Derivative	decreasingYawAngle	0.9%
7.	AmorPose4	location	landscape	0.9%
8.	AmorPose4	sees	BigChestnut_T01	0.8%
9.	AmorPose4	sees	Factory_B05	0.8%
10.	AmorPose4	sees	LimeTree_T07	0.8%
11.	AmorPose4	sees	Postoffice_B04	0.8%
12.	AmorPose4	time	16sec	0.8%
13.	AmorPose4	moves To	AmorPose5	0.8%
14.				0.1%

Fig. 5.30: A question regarding the images taken by the video camera at of AMOR-Pose4.


localhost:4567

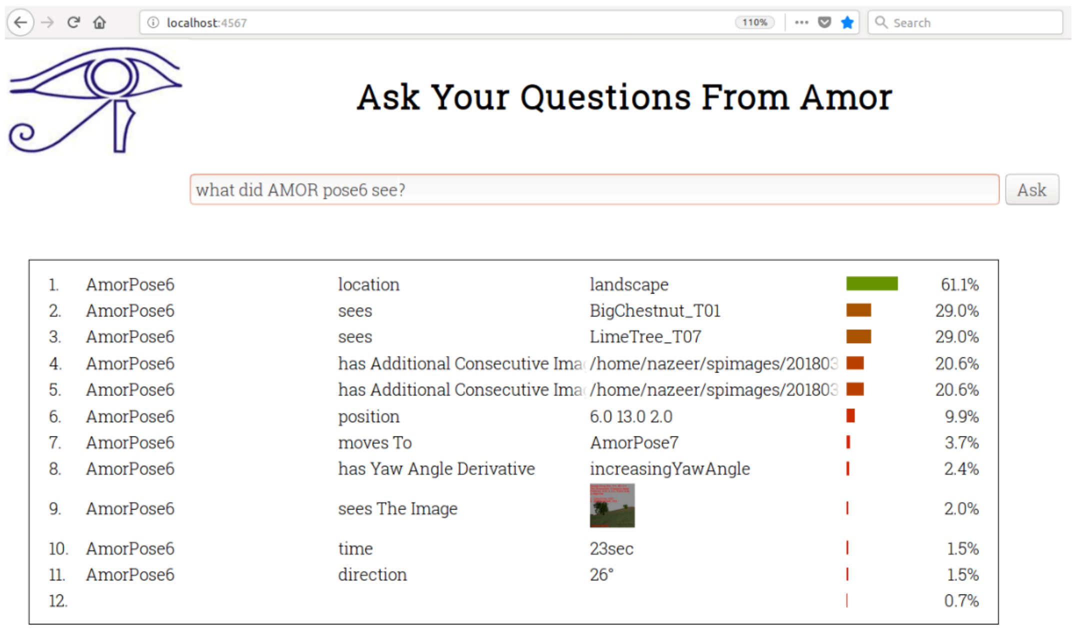
Ask Your Questions From Amor

what does AMOR Pose6 sees?

Ask

1.	AmorPose6	sees	BigChestnut_T01	🟩	62.0%
2.	AmorPose6	sees	LimeTree_T07	🟩	62.0%
3.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	🟧	30.7%
4.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	🟥	14.7%
5.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	🟥	9.0%
6.	AmorPose6	direction	26°	🟥	7.8%
7.	AmorPose6	moves To	AmorPose7	🟥	7.8%
8.	AmorPose6	position	6.0 13.0 2.0	🟥	3.1%
9.	AmorPose6	sees The Image		🟥	1.7%
10.	AmorPose6	time	23sec	🟥	1.3%
11.	AmorPose6	location	landscape	🟥	1.1%
12.				🟥	0.7%

Fig. 5.31: A question regarding the detected landmark by the semantic camera at AMOR-Pose6, asking "what does AMOR Pose6 sees", the system return the correct answer with the rank of 62.0% .


localhost:4567

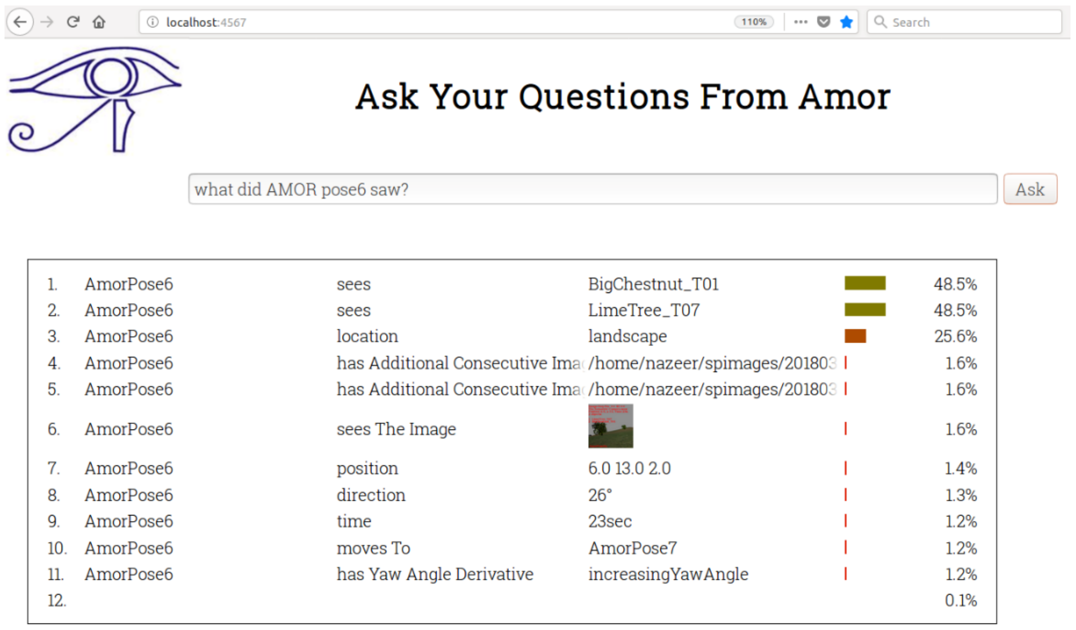
Ask Your Questions From Amor

what did AMOR pose6 see?

Ask

1.	AmorPose6	location	landscape	🟩	61.1%
2.	AmorPose6	sees	BigChestnut_T01	🟧	29.0%
3.	AmorPose6	sees	LimeTree_T07	🟧	29.0%
4.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	🟥	20.6%
5.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	🟥	20.6%
6.	AmorPose6	position	6.0 13.0 2.0	🟥	9.9%
7.	AmorPose6	moves To	AmorPose7	🟥	3.7%
8.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	🟥	2.4%
9.	AmorPose6	sees The Image		🟥	2.0%
10.	AmorPose6	time	23sec	🟥	1.5%
11.	AmorPose6	direction	26°	🟥	1.5%
12.				🟥	0.7%

Fig. 5.32: A question regarding the detected landmark by the semantic camera at AMOR-Pose6, asking "what does AMOR Pose6 sees", the system return the incorrect answer.



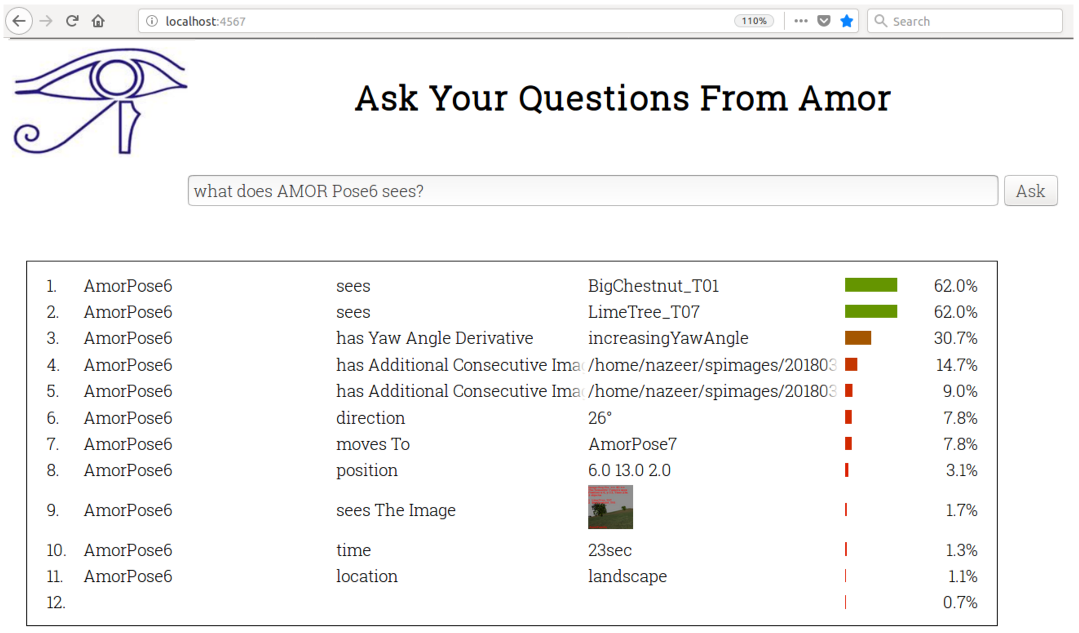
localhost:4567

Ask Your Questions From Amor

what did AMOR pose6 saw? Ask

1.	AmorPose6	sees	BigChestnut_T01	48.5%
2.	AmorPose6	sees	LimeTree_T07	48.5%
3.	AmorPose6	location	landscape	25.6%
4.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.6%
5.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	1.6%
6.	AmorPose6	sees The Image		1.6%
7.	AmorPose6	position	6.0 13.0 2.0	1.4%
8.	AmorPose6	direction	26°	1.3%
9.	AmorPose6	time	23sec	1.2%
10.	AmorPose6	moves To	AmorPose7	1.2%
11.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	1.2%
12.				0.1%

Fig. 5.33: A question regarding the detected landmark by the semantic camera at AMOR-Pose6, asking "what does AMOR Pose6 sees", the system return the correct answer with the rank of 48.5% .



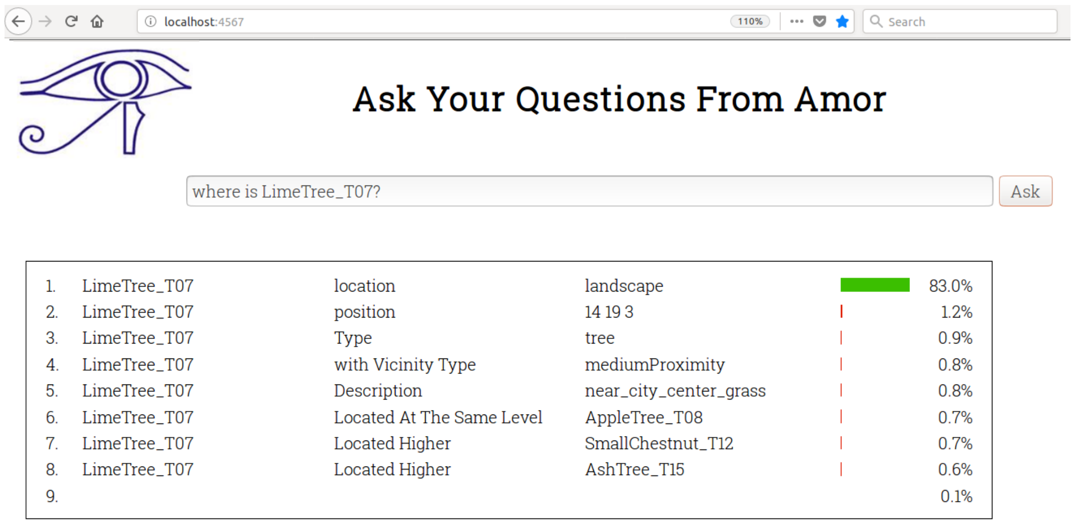
localhost:4567

Ask Your Questions From Amor

what does AMOR Pose6 sees? Ask

1.	AmorPose6	sees	BigChestnut_T01	62.0%
2.	AmorPose6	sees	LimeTree_T07	62.0%
3.	AmorPose6	has Yaw Angle Derivative	increasingYawAngle	30.7%
4.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	14.7%
5.	AmorPose6	has Additional Consecutive Ima	/home/nazeer/spimages/20180	9.0%
6.	AmorPose6	direction	26°	7.8%
7.	AmorPose6	moves To	AmorPose7	7.8%
8.	AmorPose6	position	6.0 13.0 2.0	3.1%
9.	AmorPose6	sees The Image		1.7%
10.	AmorPose6	time	23sec	1.3%
11.	AmorPose6	location	landscape	1.1%
12.				0.7%

Fig. 5.34: A question regarding the detected landmark by the semantic camera at AMOR-Pose6.

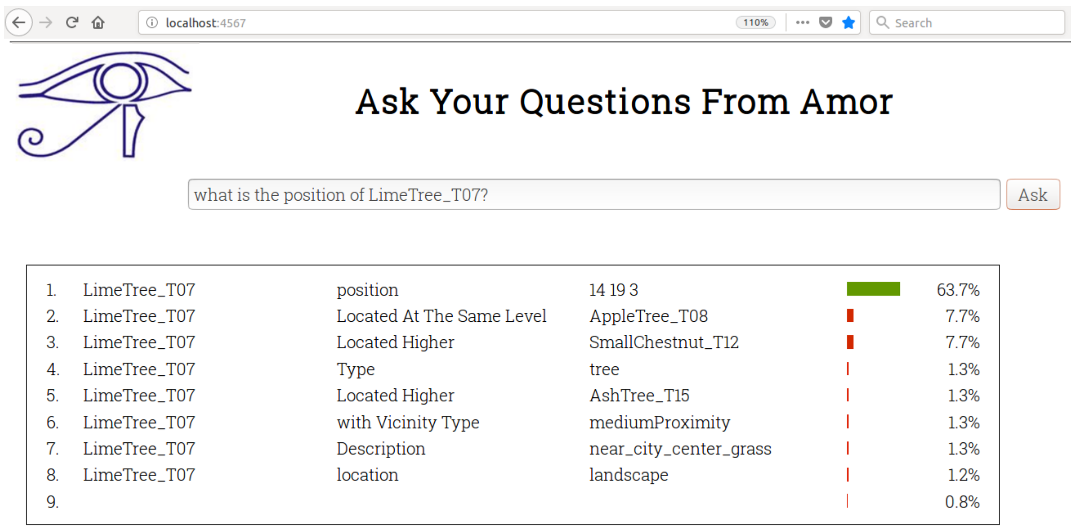


Ask Your Questions From Amor

where is LimeTree_T07?

1.	LimeTree_T07	location	landscape	🟩	83.0%
2.	LimeTree_T07	position	14 19 3	🟥	1.2%
3.	LimeTree_T07	Type	tree	🟥	0.9%
4.	LimeTree_T07	with Vicinity Type	mediumProximity	🟥	0.8%
5.	LimeTree_T07	Description	near_city_center_grass	🟥	0.8%
6.	LimeTree_T07	Located At The Same Level	AppleTree_T08	🟥	0.7%
7.	LimeTree_T07	Located Higher	SmallChestnut_T12	🟥	0.7%
8.	LimeTree_T07	Located Higher	AshTree_T15	🟥	0.6%
9.					0.1%

Fig. 5.35: A question regarding the location of the LimeTree landmark.

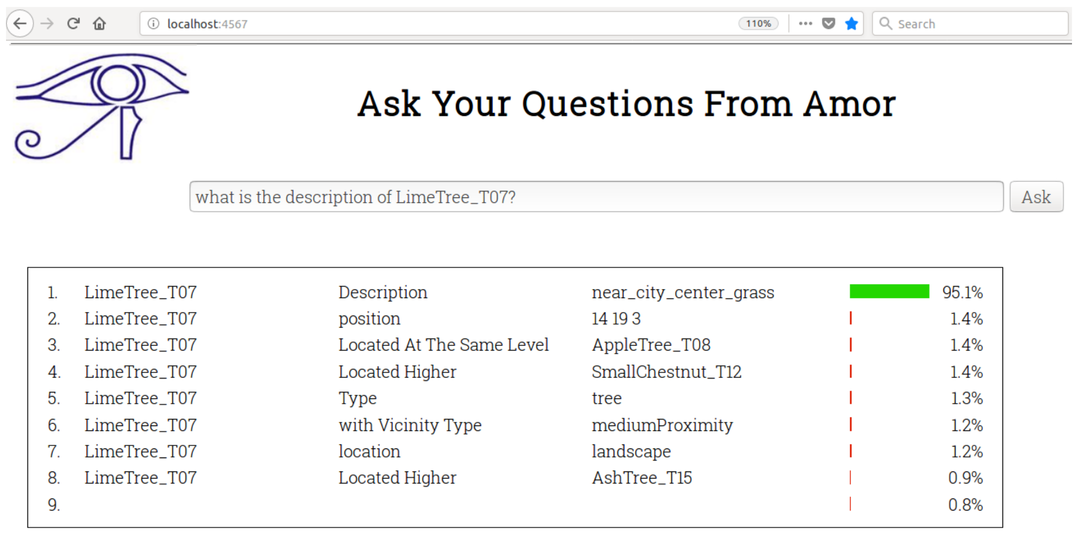


Ask Your Questions From Amor

what is the position of LimeTree_T07?

1.	LimeTree_T07	position	14 19 3	🟩	63.7%
2.	LimeTree_T07	Located At The Same Level	AppleTree_T08	🟥	7.7%
3.	LimeTree_T07	Located Higher	SmallChestnut_T12	🟥	7.7%
4.	LimeTree_T07	Type	tree	🟥	1.3%
5.	LimeTree_T07	Located Higher	AshTree_T15	🟥	1.3%
6.	LimeTree_T07	with Vicinity Type	mediumProximity	🟥	1.3%
7.	LimeTree_T07	Description	near_city_center_grass	🟥	1.3%
8.	LimeTree_T07	location	landscape	🟥	1.2%
9.				🟥	0.8%

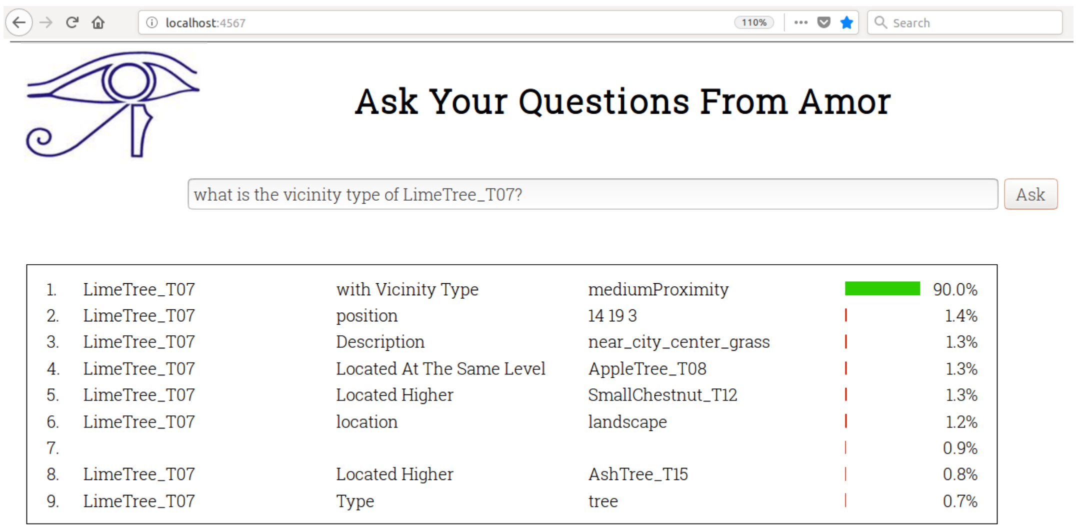
Fig. 5.36: A question regarding the position of the LimeTree landmark.



what is the description of LimeTree_T07?

1.	LimeTree_T07	Description	near_city_center_grass	🟩	95.1%
2.	LimeTree_T07	position	14 19 3		1.4%
3.	LimeTree_T07	Located At The Same Level	AppleTree_T08		1.4%
4.	LimeTree_T07	Located Higher	SmallChestnut_T12		1.4%
5.	LimeTree_T07	Type	tree		1.3%
6.	LimeTree_T07	with Vicinity Type	mediumProximity		1.2%
7.	LimeTree_T07	location	landscape		1.2%
8.	LimeTree_T07	Located Higher	AshTree_T15		0.9%
9.					0.8%

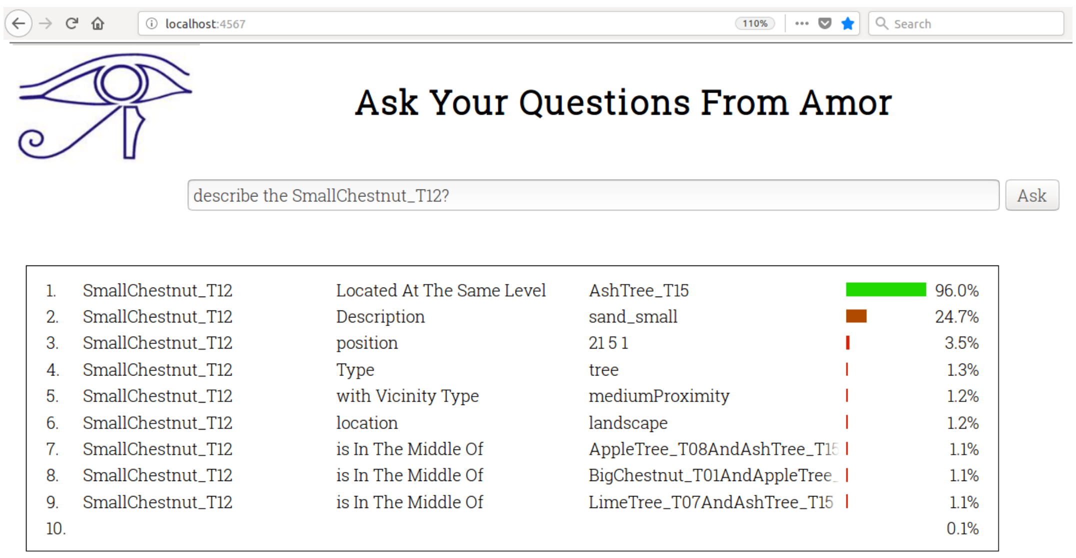
Fig. 5.37: A question regarding the description of the LimeTree landmark.



what is the vicinity type of LimeTree_T07?

1.	LimeTree_T07	with Vicinity Type	mediumProximity	🟩	90.0%
2.	LimeTree_T07	position	14 19 3		1.4%
3.	LimeTree_T07	Description	near_city_center_grass		1.3%
4.	LimeTree_T07	Located At The Same Level	AppleTree_T08		1.3%
5.	LimeTree_T07	Located Higher	SmallChestnut_T12		1.3%
6.	LimeTree_T07	location	landscape		1.2%
7.					0.9%
8.	LimeTree_T07	Located Higher	AshTree_T15		0.8%
9.	LimeTree_T07	Type	tree		0.7%

Fig. 5.38: A question regarding the vicinity type of the LimeTree landmark.



1.	SmallChestnut_T12	Located At The Same Level	AshTree_T15	96.0%
2.	SmallChestnut_T12	Description	sand_small	24.7%
3.	SmallChestnut_T12	position	21 5 1	3.5%
4.	SmallChestnut_T12	Type	tree	1.3%
5.	SmallChestnut_T12	with Vicinity Type	mediumProximity	1.2%
6.	SmallChestnut_T12	location	landscape	1.2%
7.	SmallChestnut_T12	is In The Middle Of	AppleTree_T08AndAshTree_T1	1.1%
8.	SmallChestnut_T12	is In The Middle Of	BigChestnut_T01AndAppleTree_	1.1%
9.	SmallChestnut_T12	is In The Middle Of	LimeTree_T07AndAshTree_T15	1.1%
10.				0.1%

Fig. 5.39: A question regarding the determined inter-object relation between the LimeTree and its related landmarks.

5.7 System evaluation

5.7.1 Evaluation of the AMOR core ontology

Due to their flexibility and functionality for semantic knowledge representation, ontologies have recently become the fundamental solution and are widely used for semantic and semi-semantic-driven modeling. A vast number of ontologies have been created with different sizes, complexities, and designs. They are used in diverse academic and industrial application. This diversity requires methods and mechanisms for ensuring the functionality and performance of ontologies and their capability to interact with each other within or among different platforms. This is achievable by ontology evaluation in all phases starting from its construction, release and also change or reuse (Shen et al., 2018; Bilgin et al., 2014).

Ontology evaluation is the professional analysis of an ontology concerning its frame of reference throughout every phase of its life cycle phases. Ontology evaluation is hence crucial to the development as well as the adoption of ontologies in the industry and also for academic applications (Gomez-Perez, 2001). According to Haase and Stojanovic (2005) ontology evaluation is “the timely adaptation of an ontology to the arisen changes and the consistent management of these changes” (Haase and Stojanovic, 2005, p. 199). Acknowledging that ontologies are semantically oriented, created in heterogeneous sources and are in rapid growth in both size and quantity, the process of evaluation of ontology is therefore not a straightforward but a rather challenging task (Fahad and Abdul Qadir, 2008).

According to Gomez-Perez (2001), the ontology evaluation is the means of ensuring that an ontology has been built correctly by guaranteeing that all definitions are implemented with respect to the requirements and are performing the real-world model formally and correctly. The goal is to determine what is defined and can be inferred correctly from an ontology and what is not defined or defined incorrectly and can not be inferred or might be inferred incorrectly. Gomez-Perez states a number of criteria for ontology evaluation; the proposed criteria are as follows:

- Completeness: It is rather challenging to determine the completeness of any given ontology or its definitions. Consequently, the completeness phase examines and determines the incompleteness of an ontology and its definitions.

- Conciseness: refers to the process of ensuring that the ontology does not store redundant or unnecessary definitions.

- Expandability: refers to the possibility of expanding an ontology with new definitions without modifying the defined properties.

- Consistency: This phase ensures obtaining consistent conclusions from valid input definitions.

- Sensitiveness: refers to the sensitivity of the defined properties when changes in definition occur.

In the year 2007, Obrst et al. proposed three main criteria for ontology evaluation these criteria are:

- Quality criteria: include the quality of an ontology's coverage, consistency and completeness as well as its application and performance in different scenarios and use-cases.

- Validation and verification: include the verification and validation of the structure, functionality and usability of an ontology.

- Questions with philosophical foundations: refers to the compatibility of an ontology to its upper ontology and its capability of answering the underlying philosophical theory about reality.

Based on the studies of Obrst et al. (2007) and Gomez-Perez (2001), Vrandečić introduced a comprehensive proposal for ontology evaluation. In Vrandečić (2009) he proposed the following criteria to be considered for the ontology evaluation process:

- Adaptability: refers to the ontology's functionality and flexibility with different applications and changes.

- Accuracy: refers to the correct axioms representation of the real-world aspect.

- Completeness: refers to the ability of an ontology to cover its domain of interest.

- Clarity: relates to the clarity of the ontology's definitions.

- Computational efficiency: refers to the ability of reasoning of an ontology.

- Consistency: refers to the consistency or coherence of an ontology.

- Organizational fitness: refers to the ontology's adoption given organizations.

- Conciseness: refers to the unavailability of redundant or unnecessary axioms.

An example of an application of ontology evaluation is in the work of Machová et al. (2016), where ontology evaluation is used for navigating users in a large-scale ontology space and helping them to choose the best ontology to suit their needs. Machová et al. optimized the ontology visualization with the combination of evaluation methods to expose the degree of the semantic similarity between different ontologies (Machová et al., 2016).

According to ontology engineering, the AMOR core ontology has been evaluated, and this evaluation is shown in Table 5.3.

criteria	remark
Adaptability	The AMOR core ontology is a general ontology whose core classes have been inherited from the SUMO and CORA ontology, designed for representing the navigation information. Therefore, the AMOR core ontology can be adapted and used by different mobile robots for indoor or outdoor navigation. However, depending on the environment new relations may need to be modeled.
Clarity	The AMOR core ontology has used the general formal definitions for defining terms and classes. However, there is no formula or tool to calculate or evaluate the clarity of an ontology. Consequently, it is difficult to evaluate the clarity of the AMOR core ontology.
Computational efficiency and consistency	The AMOR core ontology has been designed with the Protege software, and its consistency has been checked with hermiT 1.3.8.143 and also FaCT++ 1.6.5 reasoners. However, the relations defined for modeling the RDF-statements that represents the tour of AMOR and its semantic vision are part of the OWL full domain. This provides the flexibility in modeling relations and enables the use of the complete set of RDFS classes and properties, and also modeling RDF-statements such as blank-nodes and reification statements. However, as described in section 2.5.2.2, one limitation of using OWL full is its cost of computation and reasoning.
Organizational fitness	The application of the AMOR core ontology for modeling the tour of AMOR and also a semantic description of its environment has been tested. The results are sound proof of the fitness of the AMOR core ontology in representing the tour of AMOR and also its vision and perception of its environment.
Conciseness	The AMOR core ontology is populated from the sensory input of a simulated mobile robot. To avoid duplicated or redundant data, the first step in processing data is to filter and eliminate the duplicate and unnecessary data. As a result the AMOR core ontology does not include any redundant or unnecessary information.
Expandability	The AMOR core ontology provides general classes and depending on the application they can be extended with more specific subclasses and used to serve different purposes. Integration of ENVO ontology to the AMOR core ontology is a good example.

Tab. 5.3: The evaluation of the AMOR core ontology according to the evaluation principals of ontology engineering.

5.7.2 Evaluation of the natural language communication

During the navigation of AMOR in its environment, and more precisely while exploring the landscape and the city, the RoboSemProc populated the AMOR core ontology with a number of RDF statements that represent the tour of AMOR and the description of its environment. In section 5.6, the process of using the AMOR core ontology for natural language communication has been described; a number of examples show the functionality of the system for answering the natural language questions.

As described in section 2.8, the YodaQA system is considered as one of the most reliable and accurate question answering systems available with 80% accuracy. This section evaluates the performance of the natural language communication which is based on the YodaQA system. The evaluation has been conducted based on a number of criteria as shown in Table 5.4.

criteria	remark
Accuracy	The system is a able to answer more than 80% of direct questions. However, one of its limitations is its inability to answer indirect questions.
Speed	Depending on the size of the data sources the time needed for answering a question varies between a few seconds to one minute.
Answer style	Each candidate answer is provided with a rank value, and the highest ranked candidate answer is considered as the final answer. However, all other candidate answers are shown to the user which provides the possibility for the user to see and check other possible answers.

Tab. 5.4: The evaluation of the natural language communication.

5.7.3 Evaluation of RoboSemProc

The two main parts of the RoboSemProc are the AMOR core ontology and natural language communication. In section 5.7.1 and section 5.7.2, an evaluation of both the AMOR core ontology and the natural language communication is provided. This section provides an overview evaluation of the process of data collection.

criteria	remark
Number of detectable objects at each pose	Unlimited.
Number of pose nodes	Unlimited.
Detection of moving objects	Possible.
Defining inter-object relations	Provided. Different approaches have been implemented for molding inter-object relations between detected objects.
Sensor integration	It has been implemented and it is possible to add new sensors to the system.
Change of platform	Possible. The system can be implemented on other mobile robot platforms.
Change of environment	Possible. The system can be implemented in indoor or outdoor environments with constrain of having the semantic camera or an alternative sensor or system for object detection.
Querying collected data	Possible. With SPARQL queries and a graphical query.
Exporting RDF statements	Implemented.
Accuracy	The pose sensor is publishing 60 tics per seconds. Therefore, if the robot moves 60 meters per second which is equivalent to 216km/h, the system is able to record one pose per meter.

Tab. 5.5: The evaluation of the data collection process.

The system has been tested with the following hardware 4GB of RAM, Intel(R) Core(TM) i7-3520M @ 2.90GHz CPU with a cache size of 4096KB. The RoboSemProc operates with the MORSE open-robot and ROS system which are operating on the Linux operating system. Therefore, the system is not operating on a Windows operating system. The performance of the system under the mentioned requirements have been tested, and an overview of the hardware usage and system performance is shown in the following.

For the QA system as can be seen in Fig.5.40 the memory used with the QA system is under 45%; however, the system is developed as a single thread system and utilizes a single core of the CPU, the system monitor shows that the QA system uses 99.0% of the fourth core of the CPU to answer one question. Overall the QA system requires few seconds into a minute depending on the ontology size and the question asked to return the answer.

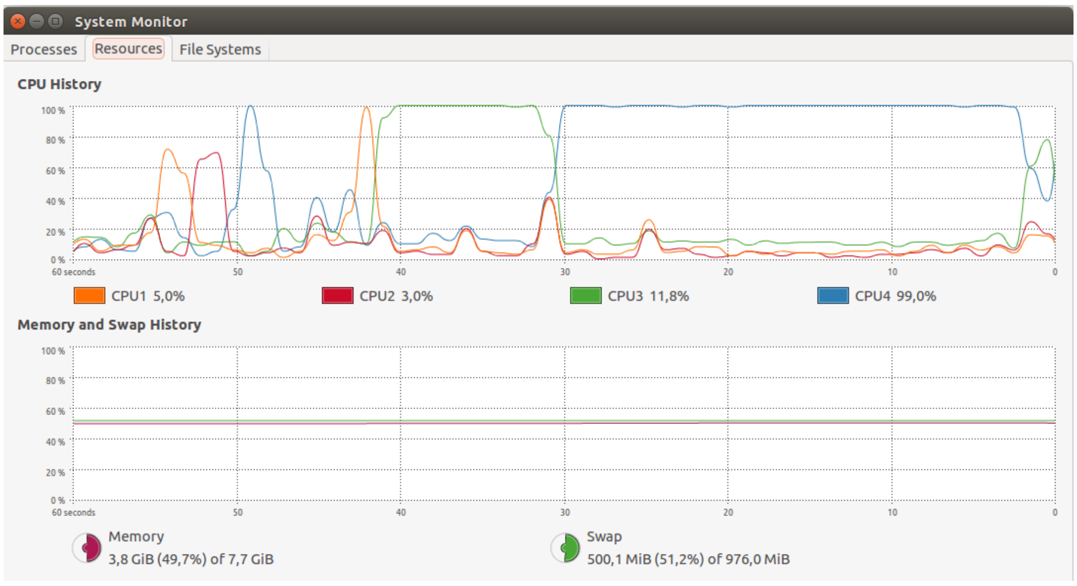
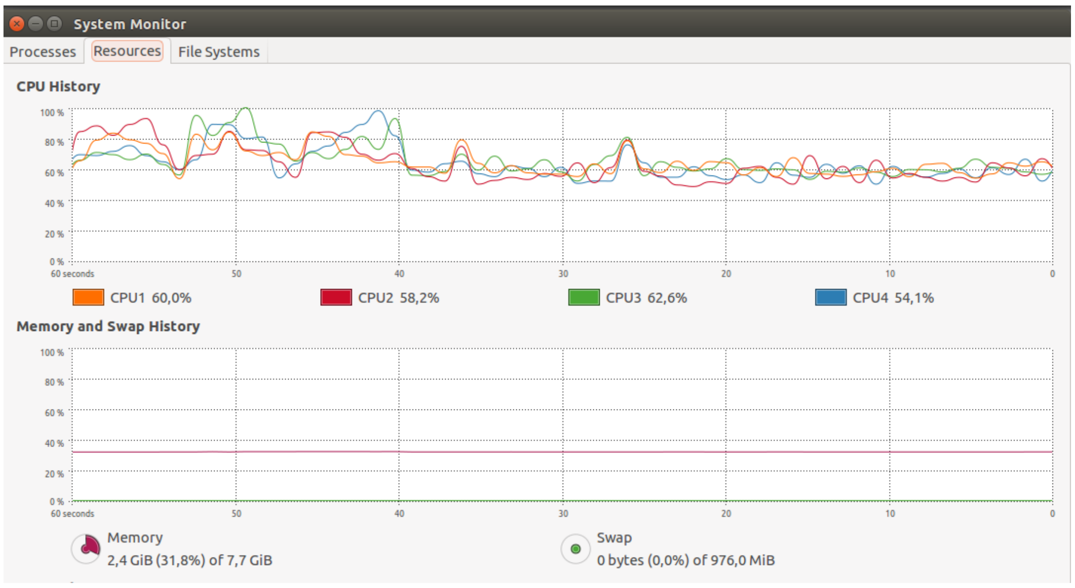


Fig. 5.40: Illustration of the performance of the mentioned hardware for running the QA system.

The performance of the system for the real-time data collection and graph visualization are shown in Fig.5.41 and Fig.5.42 accordingly. As can be seen, both systems are utilizing all cores of the CPU and depending on the size of the data the CPU usage for data collection and RDF-Graph vitalization is varying between 50% to 80%. The system does not require a large amount of memory, as shown in Fig.5.41 and Fig.5.42 the memory usage for both data collection and visualization is below 50% of the mentioned memory.

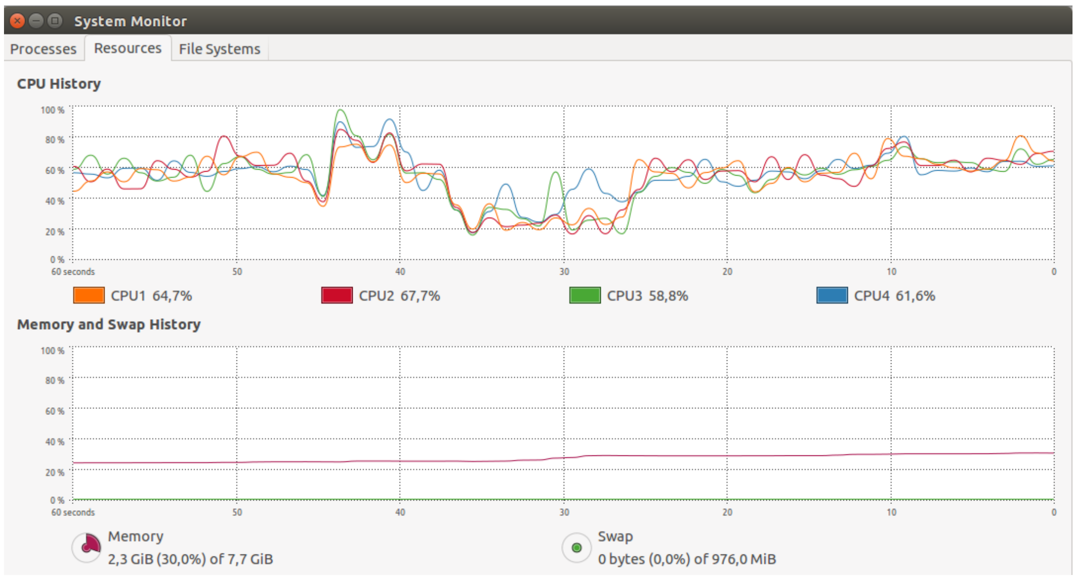


🟧 CPU1 60,0% 🟥 CPU2 58,2% 🟩 CPU3 62,6% 🟦 CPU4 54,1%

Memory
2,4 GiB (31,8%) of 7,7 GiB

Swap
0 bytes (0,0%) of 976,0 MiB

Fig. 5.41: Illustration of the performance of the mentioned hardware for the navigation and data collection.



🟧 CPU1 64,7% 🟥 CPU2 67,7% 🟩 CPU3 58,8% 🟦 CPU4 61,6%

Memory
2,3 GiB (30,0%) of 7,7 GiB

Swap
0 bytes (0,0%) of 976,0 MiB

Fig. 5.42: Illustration of the performance of the mentioned hardware graph visualization.

Conclusion 6

The only constant in the technology industry is change.

— **Marc Benioff**
(author and philanthropist)

6.1 Summary

The motivation of this research was to explore ways of managing sensory data from mobile robots to provide them with the possibility of describing their explored environment semantically and moreover, to use the resulting semantic information for communication of information, and, finally, promote natural language communication with human beings.

This research has drawn on considerations of different fields in the context of semantic description and communication, such as semantic knowledge acquisition from sensory data and semantic knowledge representation, robot-robot and human-robot communication and natural language communication. As a result, a novel system called RoboSemProc has been proposed; the RoboSemProc enables a mobile robot to use its sensory output to create semantic information representing its tour and the explored environment and to use it for the communication of information. This system provides approaches for modeling information in a formal language that can be understood by robots and used for communication and interaction between them; it is also structured in a way that humans can interpret, understand and use the information to communicate to robots in a natural language.

In this context, the following sections address the main contributions and the outlook of this dissertation research.

6.2 The main contributions

To maintain and evaluate findings, each stage of the research has been backed by experiments, which provide feedback on how to enhance the application of the RoboSemProc. The result of the experiments is a proof of the soundness of the approach to model semantic information, using a combination of different sensors from a mobile robot, and applying that information for communication of information.

The major contributions of this research are as follows:

- Design and modeling of a well-defined, tailored ontology called the AMOR core ontology, that can be used for the representation of the tour of mobile robots as well as their explored environment, in a human and robot interpretable way.
- Introducing a novel approach, called the helperNode, for modeling n-ary RDF relations that represent inter-object relations between landmarks within the environment.
- Semantic knowledge acquisition from sensory data of different sensors to create single RDF relations.
- Automatic ontology instantiation of human and machine interpretable semantic triples in real-time during navigation of a mobile robot.
- Tailoring of a natural language processing enhanced QA system for implementing a natural language interface for human robot communication.

6.3 Outlook

The RoboSemProc has been designed with the intention of future extensions; the system has been customized for the use of navigation of AMOR in its simulated outdoor environment. The AMOR core ontology, however, is a well-defined, tailored ontology that can be extended or customized for different mobile applications, such as indoor navigation or multi-robot navigation. Depending on the use and application the system can be extended with new relations to form new RDF-statements.

Future extensions can enable the system to detect moving items and also to populate the AMOR core ontology in real-time by multiple robots simultaneously. The communication system provided by the YodaQA system also has room for improvements. For instance, answering indirect questions, yes and no questions and also listing questions can be considered areas for future development.

Publications

1. Nazeer T Mohammed Saeed, Madjid Fathi, Klaus-Dieter Kuhnert, "An Approach for Instant Conversion of Sensory Data of a Simulated Sensor of a Mobile Robot into Semantic Information". In: 2018 IEEE International Conference on Electro/Information Technology, pp. 0142-0146, 2018, USA.

2. Nazeer T Mohammed Saeed, Christian Weber, Madjid Fathi, Klaus-Dieter Kuhnert, "An Approach for Modeling Spatial Prepositions with RDF Reification and Blank Nodes Based on the Environment Perception of a Simulated Mobile Robot". In: 61st IEEE International Midwest Symposium on Circuits and Systems, pp. 713-716, 2018, Canada.

3. Nazeer T Mohammed Saeed, Christian Weber, Madjid Fathi, Klaus-Dieter Kuhnert, "An Efficient Alternative for Modeling Spatial Prepositions with RDF Helper Nodes Based on the Environment Perception of a Mobile Robot". In: 28th IEEE International Symposium on Industrial Electronics, 2019, Canada.

4. Nazeer T Mohammed Saeed, Christian Weber, Ahlam Mallak, Madjid Fathi, Roman Obermaisser, Klaus-Dieter Kuhnert, "ADISTES Ontology for Active Diagnosis of Sensors and Actuators in Distributed Embedded Systems". In: 19th IEEE International Conference on Electro/Information Technology, 2019, USA.

5. Nazeer T Mohammed Saeed, Masoud Fathi, Madjid Fathi, Klaus-Dieter Kuhnert, “Robot Semantic Protocol (RoboSemProc) for Semantic Environment Description and Human-Robot Communication. In: International Journal of Social Robotics, Springer, 2019 (under review).

6. Masoud Fathi Kazerouni, Nazeer T Mohammed Saeed, Klaus-Dieter Kuhnert, "Fully-automatic Natural Plant Recognition System Using Deep Neural Network for Dynamic Outdoor Environments". In: Pattern Analysis and Applications, Springer, 2019 (under review).

.

Bibliography

Almendra, Vinicius and Daniel Schwabe (2006). „Trust policies for semantic web repositories". In: *Proc. 2nd Int. Semant. Web Policy Work.* Pp. 17–31 (cit. on p. 27).

Antoniou, Grigoris and Frank van Harmelen (2004). *A Semantic Web Primer*. MIT press (cit. on p. 35).

— (2009). „Web Ontology Language: OWL". In: *Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg*. Springer Berlin Heidelberg, pp. 91–110 (cit. on pp. 36–38).

Barrera, Alejandra (2011). *Advances in Robot Navigation*. Ed. by Alejandra Barrera. InTech, p. 10 (cit. on p. 10).

Barrera, Alejandra, Jessica Lindblom, Rebecca Andreasson, and Tom Ziemke (2017). „User Experience in Social Human-Robot Interaction". In: *Int. J. Ambient Comput. Intell.* 8.2, pp. 12–31 (cit. on pp. 13, 14).

Bast, Hannah and Elmar Haussmann (2015). „More accurate question answering on freebase". In: *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag. ACM* (cit. on p. 102).

Baudiš, Petr and Jan Šedivý (2015). „Modeling of the question answering task in the yodaqa system". In: *Int. Conf. Cross-Language Eval. Forum Eur. Lang.* (Cit. on pp. 45, 46).

Berners-Lee, Timothy John (2001). „The Semantic Web". In: *Sci. Am.* 284, pp. 34–43 (cit. on pp. 18, 19, 34).

Berners-Lee, Timothy John, Robert Cailliau, and Ari Luotonen (1994). „p76-berners-lee.pdf". In: *Commun. ACM* 37, p. 8 (cit. on p. 22).

Bilgin, Gozde, Irem Dikmen, and M. Talat Birgonul (2014). „Ontology Evaluation: An Example of Delay Analysis". In: *Procedia Eng.* 85, pp. 61–68 (cit. on p. 134).

Bräunl, Thomas (2008). „Localization and Navigation," in: *Embed. Robot.* Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 241–269 (cit. on p. 10).

Brenda, Maria (2014). *The cognitive perspective on the polysemy of the English spatial preposition over*. Cambridge Scholars Publishing (cit. on p. 85).

Buneman, Peter, Adriane Chapman, and James Cheney (2006). „Provenance management in curated databases". In: *2006 ACM SIGMOD Int. Conf. Manag. data*, pp. 539–550 (cit. on p. 27).

Callahan, Alison and Michel Dumontier (2013). „Ovopub: Modular data publication with minimal provenance". In: *arXiv:1305.6800*. arXiv: `1305.6800` (cit. on pp. 27, 89).

Carbonera, Joel Luis, Sandro Rama Fiorini, Edson Prestes, et al. (2013). „Defining positioning in a core ontology for robotics“. In: *2013 IEEE/RSJ Int. Conf. Intell. Robot. Syst.* Pp. 1867–1872 (cit. on pp. 3, 41).

Carroll, Jeremy J., Christian Bizer, Pat Hayes, and Patrick Stickler (2005). „Named graphs“. In: *Web Semant. Sci. Serv. Agents World Wide Web* 3.4, pp. 247–267 (cit. on p. 28).

Chen, Lei, Haifei Zhang, Ying Chen, and Wenping Guo (2012). „Blank Nodes in RDF.“ In: *J. Software, SWJ* 7.9, pp. 1993–1999 (cit. on p. 26).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille (2014). „Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs“. In: *arXiv Prepr. arXiv1412.7062*, pp. 1–14. arXiv: 1412.7062 (cit. on pp. 15, 52).

Clarke, Roger (1993). „Asimov's laws of robotics: implications for information technology-Part I“. In: *Computer (Long. Beach. Calif).* 26.12, pp. 53–61 (cit. on p. 8).

Compton, Michael, Cory Henson, Laurent Lefort, Holger Neuhaus, and Amit Sheth (2009a). *A Survey of the Semantic Specification of Sensors*. Tech. rep. (cit. on p. 40).

— (2009b). *A Survey of the Semantic Specification of Sensors*. Tech. rep. (cit. on pp. 42, 60).

Dautenhahn, Kerstin (2007). „Socially intelligent robots: Dimensions of human-robot interaction“. In: *Philos. Trans. R. Soc. B Biol. Sci.* 362.1480, pp. 679–704 (cit. on pp. 13, 14).

Dietze, Stefan and John Domingue (2009). „Bridging between sensor measurements and symbolic ontologies through conceptual spaces“. In: *Int. Work. Semant. Sens. Web (SemSensWeb 2009) 6th Annu. Eur. Semant. Web Conf.* (Cit. on p. 40).

Duarte, Miguel, Fernando Silva, Tiago Rodrigues, Sancho Moura Oliveira, and Anders Christensen (2014). „JBotEvolver: A Versatile Simulation Program for Evolutionary Robotics“. In: *Artif. Life 14 Proc. Fourteenth Int. Conf. Synth. Simul. Living Syst.* Alife 14, pp. 210–211 (cit. on pp. 17, 50).

Eckert, Jr John Presper and John W. Mauchly (1964). „Electronic numerical integrator and computer“. In: *Electron. Numer. Integr. Comput.* (Cit. on p. 9).

Epstein, Edward A., Marshall I. Schor, and Bhavani S. Iyer (2012). „Making Watson fast“. In: *IBM J. Res. Dev.* (Cit. on p. 46).

Everingham, Mark, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman (2012). „The Pascal Visual Object Classes (VOC) Challenge“. In: *Int. J. Comput. Vis.* 88, pp. 303–338 (cit. on p. 16).

Fahad, Muhammad and Muhammad Abdul Qadir (2008). „A Framework for Ontology Evaluation.“ In: *ICCS Suppl.* 149-158 (cit. on p. 134).

Ferguson, Dave and Anthony Stentz (2006). „Using Interpolation to Improve Path Planning : The Field D * Algorithm“. In: *J. F. Robot.* 23.2, pp. 79–101 (cit. on p. 11).

Ferrucci, David, Eric Nyberg, James Allan, and Ken Barker (2009). „Towards the open advancement of question answering systems“. In: *IBM, Armonk, NY, IBM Res. Rep* (cit. on p. 45).

Filliat, David, Emmanuel Battesti, Stéphane Bazeille, et al. (2012). „RGBD object recognition and visual texture classification for indoor semantic mapping“. In: *2012 IEEE Conf. Technol. Pract. Robot Appl. TePRA 2012*, pp. 127–132 (cit. on pp. 14, 52).

Fiorini, Sandro Rama, Julita Bermejo-Alonso, Paulo Goncalves, and Edison Pignaton de Freitas (2017). „A suite of ontologies for robotics and automation“. In: *IEEE Robot. Autom. Mag.* 24, pp. 8–11 (cit. on pp. 41, 60, 63).

Furche, Tim, Benedikt Linse, Francois Bry, Dimitris Plexousakis, and Georg Gottlob (2006). „RDF querying: Language constructs and evaluation methods compared“. In: *Reason. Web* (cit. on pp. 31, 32).

Gandon, Fabien, Reto Krummenacher, Sung-Kook Han, and Ioan Toma (2011). *The Resource Description Framework and its Schema. Handbook of Semantic Web Technologies*, pp. 978–981 (cit. on p. 23).

Ganz, Frieder, Payam Barnaghi, and Francois Carrez (2016). „Automated Semantic Knowledge Acquisition from Sensor Data“. In: *IEEE Syst. J.* 10.3, pp. 1214–1225 (cit. on pp. 39, 40).

Genesereth, Michael R. and Nils J. Nilsson (1987). „Logical foundations of artificial intelligence.“ In: *The Journal of Symbolic Logic* 55.03, pp. 1304–1307 (cit. on p. 34).

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). „Rich feature hierarchies for accurate object detection and semantic segmentation“. In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* Pp. 580–587. arXiv: `1311.2524` (cit. on pp. 15, 52).

Godwin-Jones, Robert (2010). „Emerging technologies“. In: *Lang. Learn. Technol.* (Cit. on p. 107).

Gomez-Perez, Asuncion (2001). „Evaluation of ontologies“. In: *Int. J. Intell. Syst.* 16.3, pp. 391–409 (cit. on pp. 134, 135).

Goodrich, Michael A. and Alan C. Schultz (2007). „Human-Robot Interaction: A Survey“. In: *Found. Trends® Human-Computer Interact.* 1.3, pp. 203–275. arXiv: `arXiv:1011.1669v3` (cit. on p. 12).

Groth, Paul, Andrew Gibson, and Jan Velterop (2010). „The anatomy of a nanopublication“. In: *Inf. Serv. Use* 30.1-2, pp. 51–56 (cit. on pp. 27, 89).

Gruber, Thomas R. (1993). „A translation approach to portable ontology specifications“. In: *Knowl. Acquis.* Pp. 199–220 (cit. on pp. 33, 34).

Gutierrez, Claudio, Carlos Hurtado, and Alberto O. Mendelzon (2004). „Foundations of semantic web databases“. In: *Proc. twenty-third ACM SIGMOD-SIGACT-SIGART Symp. Princ. database Syst. - Pod. '04*. New York, New York, USA: ACM Press, p. 95 (cit. on pp. 26, 89).

Guzzi, Jerome, Alessandro Giusti, Luca M. Gambardella, Guy Theraulaz, and Gianni A. Di Caro (2013). „Human-friendly Robot Navigation in Dynamic Environments“. In: *2013 IEEE Int. Conf. Robot. Autom.* Pp. 423–430 (cit. on p. 11).

Gyrard, Amelie, Martin Serrano, Joao Bosco Jares, Soumya Kanti Datta, and Muhammad Intizar Ali (2017). „Sensor-based Linked Open Rules (S-LOR)“. In: *Proc. 26th Int. Conf. World Wide Web Companion - WWW '17 Companion*. New York, New York, USA: ACM Press, pp. 1153–1159 (cit. on p. 40).

Haase, Peter and Ljiljana Stojanovic (2005). „Consistent Evolution of OWL Ontologies". In: *ESWC 2005 Semant. Web Res. Appl.* Springer, Berlin, Heidelberg, pp. 182–197 (cit. on p. 134).

Haase, Peter, Jeen Broekstra, Andreas Eberhart, and Raphael Volz (2004). „A comparison of RDF query languages". In: *ISWC 2004 Semant. Web – ISWC 2004*, pp. 502–517 (cit. on p. 30).

Hanna, Josiah P. and Peter Stone (2017). „Grounded Action Transformation for Robot Learning in Simulation". In: *Proc. Thirty-First AAAI Conf. Artif. Intell.* February, pp. 3834–3840 (cit. on pp. 17, 50).

Hartig, Olaf and Bryan Thompson (2014). „Foundations of an Alternative Approach to Reification in RDF". In: *arXiv Prepr. arXiv1406.3399*. arXiv: `1406.3399v1` (cit. on pp. 27, 28).

Heath, Tom and Christian Bizer (2011). „Linked Data: Evolving the Web into a Global Data Space". In: *Synth. Lect. Semant. Web Theory Technol.* 1.1, pp. 1–136 (cit. on pp. 26, 89).

Hitzler, Pascal, Markus Krotzsch, and Sebastian Rudolph (2009). *Foundations of semantic web technologies*. Chapman and Hall/CRC, p. 456 (cit. on pp. 34, 36, 37, 39).

Hogan, Aidan (2015). „Skolemising Blank Nodes while Preserving Isomorphism". In: *Proc. 24th Int. Conf. World Wide Web - WWW '15*. New York, New York, USA: ACM Press, pp. 430–440 (cit. on pp. 26, 89).

Hogan, Aidan, Marcelo Arenas, Alejandro Mallea, and Axel Polleres (2014). „Everything you always wanted to know about blank nodes". In: *Web Semant. Sci. Serv. Agents World Wide Web* 27-28, pp. 42–69 (cit. on p. 27).

Holsapple, Clyde W. and K. D. Joshi (2002). „A collaborative approach to ontology design". In: *Commun. ACM* 45.2, pp. 42–47 (cit. on p. 62).

Horrocks, Ian, Peter F. Patel-Schneider, and Frank van Harmelen (2003). „From SHIQ and RDF to OWL: the making of a Web Ontology Language". In: *Web Semantic Sci. Serv. Agents World Wide Web* 1.1, pp. 7–26 (cit. on p. 36).

Käfer, Tobias, Ahmaed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan (2013). „Observing linked data dynamics". In: *ESWC 2013 Semantic Web Semant. Big Data*, pp. 213–227 (cit. on pp. 26, 89).

Kehoe, Ben, Sachin Patil, Pieter Abbeel, and Ken Goldberg (2015). „A Survey of Research on Cloud Robotics and Automation". In: *IEEE Trans. Autom. Sci. Eng.* 12.2, pp. 398–409. arXiv: `arXiv:1011.1669v3` (cit. on pp. 17, 50).

Krötzsch, Markus (2012). „OWL 2 Profiles: An Introduction to Lightweight Ontology Languages". In: *Reason. Web 2012 Reason. Web. Semant. Technol. Adv. Query Answering*. Springer, Berlin, Heidelberg, pp. 112–183 (cit. on p. 39).

Kruse, Thibault, Patrizia Basili, Stefan Glasauer, and Alexandra Kirsch (2012). „Legible robot navigation in the proximity of moving humans". In: *Proc. IEEE Work. Adv. Robot. its Soc. Impacts, ARSO*, pp. 83–88 (cit. on p. 10).

Kuderer, Markus, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard (2012). „Feature-Based Prediction of Trajectories for Socially Compliant Navigation". In: *Robot. Sci. Syst. VIII* (cit. on p. 11).

Kuhnert, Klaus-Dieter (2008). „Software architecture of the Autonomous Mobile Outdoor Robot AMOR“. In: *2008 IEEE Intell. Veh. Symp.* IEEE, pp. 889–894 (cit. on p. 49).

Kuipers, Benjamin, Yung-Tai Byun, Rick Froom, et al. (2000). *The Spatial Semantic Hierarchy Spatial Semantic Hierarchy*. Tech. rep. (cit. on pp. 41, 60).

Kulkarni, Indraneel S. and Dario Pompili (2010). „Task allocation for networked autonomous underwater vehicles in critical missionsw“. In: *Sel. Areas Commun. IEEE J.* 28.5, pp. 716–727 (cit. on p. 13).

Lally, Adam (2011). „Natural language processing with prolog in the IBM Watson system“. In: *Assoc. Log. Program.* (Cit. on p. 45).

Legge, Eric L.G., Christopher R. Madan, and Enoch T. Ng (2012). „Building a memory palace in minutes: Equivalent memory performance using virtual versus conventional environments with the Method of Loci“. In: *Acta Psychol. (Amst).* (Cit. on p. 107).

Liu, Yugang and Goldie Nejat (2013). „Robotic urban search and rescue: A survey from the control perspective“. In: *J. Intell. Robot. Syst. Theory Appl.* 72.2, pp. 147–165 (cit. on pp. 12, 13).

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). „Fully convolutional networks for semantic segmentation“. In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 07-12-June, pp. 3431–3440. arXiv: 1411.4038 (cit. on pp. 15, 52).

Losey, Dylan P., Craig G. McDonald, Edoardo Battaglia, and Marcia K. O’Malley (2018). „A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical Human–Robot Interaction“. In: *Appl. Mech. Rev.* 70.1, p. 010804. arXiv: arXiv:1706.00155 (cit. on pp. 2, 12).

Machová, Kristína, Jozef Vrana, Marián Mach, and Peter Sinčák (2016). „Ontology evaluation based on the visualization methods, context and summaries“. In: *Acta Polytech. Hungarica* 13.4, pp. 53–76 (cit. on p. 136).

Mallak, Ahlam, Christian Weber, and Madjid Fathi (2017). „Active diagnosis automotive ontology for distributed embedded systems“. In: *Technol. Eng. Manag. Summit* (cit. on p. 62).

Marin, Draltan (2006). „A note on the history of the document : RDF Formalization“. In: *Comput. Sci. Dep. Univ. Chile*. May (cit. on pp. 22–24).

Marx, Edgard, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, et al. (2014). „Towards an open question answering architecture“. In: *Proc. 10th Int. Conf. Semant. Syst. - SEM ’14*. New York, New York, USA: ACM Press, pp. 57–60 (cit. on p. 45).

Minguez, Javier and Luis Montano (2004). „Nearness Diagram (ND) Navigation : Collision Avoidance in Troublesome Scenarios“. In: *IEEE Trans. Robot. Autom.* 20.1, pp. 45–59 (cit. on p. 11).

Mohammed Saeed, Nazeer T., Christian Weber, Madjid Fathi, and Klaus-Dieter Kuhnert (2018). „An Approach for Modeling Spatial Prepositions with RDF Reification and Blank Nodes Based on the Environment Perception of a Simulated Mobile Robot“. In: *61st IEEE Int. Midwest Symp. Circuits Syst.* (Cit. on pp. 85, 86).

Nguyen, Tu Ngoc and Wolf Siberski (2013). „SLUBM: An Extended LUBM Benchmark for Stream Reasoning.“ In: *OrdRing’13 Proc. 2nd Int. Conf. Ordering Reason.* Pp. 43–54 (cit. on pp. 27, 89).

Nguyen, Vinh, Olivier Bodenreider, and Amit Sheth (2014). „Don't like RDF reification?“ In: *Proc. 23rd Int. Conf. World wide web - WWW*. New York, New York, USA: ACM Press, pp. 759–770 (cit. on pp. 27, 29).

Nguyen, Vinh, Olivier Bodenreider, Krishnaprasad Thirunarayan, et al. (2015). *On Reasoning with RDF Statements about Statements using Singleton Property Triples*. Tech. rep. arXiv: `1509.04513v1` (cit. on pp. 27, 29, 89).

Obrst, Leo, Werner Ceusters, Inderjeet Mani, Steve Ray, and Barry Smith (2007). „The Evaluation of Ontologies“. In: *Semant. Web*. Boston, MA: Springer US, pp. 139–158 (cit. on p. 135).

Paull, Liam, Gaetan Severac, and Guilherme V. Raffo (2012). „Towards an ontology for autonomous robots“. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1359–1364 (cit. on pp. 42, 60).

Pérez, Jorge, Marcelo Arenas, and Claudio Gutierrez (2009). „Semantics and complexity of SPARQL“. In: *ACM Trans. Database Syst.* 34.3, pp. 1–45 (cit. on pp. 30, 32).

Pichler, Reinhard, Axel Polleres, Fang Wei, and Stefan Woltran (2008). „dRDF: entailment for domain-restricted RDF“. In: *ESWC 2008 Semant. Web Res. Appl.* Pp. 200–2014 (cit. on pp. 26, 89).

Plauska, Ignas (2013). *Ontology for Robot Programming domain*. Tech. rep. (cit. on pp. 42, 60).

Powers, Shelley (2003a). *Practical RDF* (cit. on p. 22).

— (2003b). *Practical RDF: solving problems with the resource description framework*. O'Reilly Media, Inc (cit. on p. 27).

Pradeep, Yazhini C., Zhu Ming, Manuel Del Rosario, and Peter C.Y. Chen (2016). „Human-inspired robot navigation in unknown dynamic environments“. In: *2016 IEEE Int. Conf. Mechatronics Autom. IEEE ICMA 2016*, pp. 971–976 (cit. on p. 11).

Prestes, Edson, Sandro Rama Fiorini, and Joel Carbonera (2014). „Core ontology for robotics and automation“. In: *International Program Committee*, p. 7 (cit. on p. 2).

Qu, Zeng and Bo Yuan (2015). „Kinematics Analysis and Simulation of Foaming Mold Cleaning Robot by Matlab Robotics Toolbox“. In: *Proc. Int. Conf. Adv. Mech. Eng. Ind. Informatics* 15.Ameii, pp. 1655–1660 (cit. on pp. 17, 50).

Robotics SPARC (2016). *Robotics 2020 Multi-Annual Roadmap for Robotics in Europe*. Tech. rep. SPARC, The partnership for Robotics in Europe (cit. on p. 2).

Roehrbein, Florian, Marc Oliver Gewaltig, Cecilia Laschi, et al. (2016). „The Neurorobotic Platform: A simulation environment for brain-inspired robotcs“. In: *Int. Symp. Robot.* 2016, pp. 387–392 (cit. on pp. 17, 50).

Rosenfeld, Ariel, Noa Agmon, Oleg Maksimov, and Sarit Kraus (2017). „Intelligent agent supporting human–multi-robot team collaboration“. In: *Artif. Intell.* 252, pp. 211–231 (cit. on p. 13).

Russakovsky, Olga, Jia Deng, Hao Su, et al. (2015). „ImageNet Large Scale Visual Recognition Challenge“. In: *Int. J. Comput. Vis.* 115.3, pp. 211–252 (cit. on p. 16).

Saez-Pons, Joan, Lyuba Alboul, Jacques Penders, and Leo Nomdedeu (2010). „Multi-robot team formation control in the GUARDIANS project“. In: *Ind. Rob.* 37.4, pp. 372–383 (cit. on p. 13).

Sahoo, Satya S., Vinh Nguyen, Olivier Bodenreider, et al. (2011). „A unified framework for managing provenance information in translational research“. In: *BMC Bioinformatics* 12.1, p. 461 (cit. on pp. 27, 89).

Schätzle, Alexander, Martin Przyjaciel-Zablocki, Simon Skilevic, and Georg Lausen (2016). „S2RDF: RDF querying with SPARQL on spark“. In: *Proc. VLDB Endow.* Pp. 804–815 (cit. on p. 32).

Schlaefer, Nico, Petra Gieselmann, Thomas Schaaf, and Alex Waibel (2006). „A pattern learning approach to question answering within the ephyra framework“. In: *Int. Conf. Text, Speech and Dialogue*. Berlin: Springer, pp. 687–694 (cit. on p. 45).

Schlenoff, Craig, Edson Prestes, Raj Madhavan, et al. (2012). „An IEEE standard Ontology for Robotics and Automation“. In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst.* IEEE, pp. 1337–1342 (cit. on pp. 3, 41).

Schwarte, Christian (2017). „Building a Resource Description Framework (RDF) for Semantic Mapping by a Simulated Mobile Robot“. PhD thesis. Siegen, p. 73 (cit. on pp. 64, 69).

Shannon, Robert (1998). „Introduction To the Art and Science of Simulation“. In: *Proc. 30th Conf. Winter Simul.* Pp. 7–14. arXiv: `arXiv:1011.1669v3` (cit. on pp. 16, 50).

Shen, Ying, Daouyuan Chen, Min Yang, Yaliang Li, and Nan Du (2018). „Ontology Evaluation with Path-based Text-aware Entropy Computation“. In: *Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.* (Cit. on p. 134).

Spitkovsky, Valentin I. and Angel X. Chang (2012). *A Cross-Lingual Dictionary for English Wikipedia Concepts.* Pp. 3168–3175 (cit. on p. 102).

Stocker, Markus, Mauno Rönkkö, and Mikko Kolehmainen (2012). „Making sense of sensor data using ontology: A discussion for road vehicle classification“. In: *Int. Congr. Environ. Model. Softw.* (Cit. on p. 40).

Stückler, Jorg, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke (2015). „Dense real-time mapping of object-class semantics from RGB-D video“. In: *Real-Time Image Proc* (cit. on pp. 14, 52).

Trischler, Adam, Tong Wang, Xingdi Yuan, et al. (2016). „NewsQA: A Machine Comprehension Dataset“. In: *arXiv:1611.09830*. arXiv: `1611.09830` (cit. on pp. 44, 97).

Tzitzikas, Yannis, Christina Lantzaki, and Dimitris Zeginis (2012). „Blank node matching and RDF/S comparison functions“. In: *Semant. Web , Springer*, pp. 591–607 (cit. on pp. 26, 89).

Vrandečić, Denny (2009). „Ontology Evaluation“. In: *Handb. Ontol.* Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 293–313 (cit. on p. 135).

Whitmore, Andrew, Anurag Agarwal, and Li Da Xu (2015). „The Internet of Things—A survey of topics and trends“. In: *Inf. Syst. Front.* 17.2, pp. 261–274 (cit. on p. 39).

Yao, Xuchen (2015). „Lean question answering over freebase from scratch“. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstration*. QAlabel-lookupyao (cit. on p. 102).

Websites

Bergman, Michael K. (2018). *Metamodeling in Domain Ontologies*. URL: http://www.mkbergman.com/ (visited on Nov. 4, 2018) (cit. on pp. 37, 38).

Chen, Andy and Chaitanya Asawa (2017). *Semantic segmentation*. URL: https://thegradient.pub/semantic-segmentation/ (visited on Aug. 1, 2018) (cit. on p. 52).

European Molecular Biology (2018). *The Ontology Lookup Service, EMBL-EBI, ENVO ontology*. URL: https://www.ebi.ac.uk/ols/ontologies/envo/terms/graph?iri=http://purl.obolibrary.org/obo/ENVO{_}00000073 (visited on Dec. 13, 2018) (cit. on pp. 96, 97).

Franz Inc (2018a). *AG Semantic Graph Database*. URL: https://franz.com/agraph/allegrograph/ (visited on Dec. 5, 2018) (cit. on p. 56).

— (2018b). *AG WebView*. Online. URL: https://franz.com/agraph/support/documentation/ (visited on Dec. 5, 2018) (cit. on p. 56).

— (2018c). *Gruff: A Grapher-Based Triple-Store Browser*. URL: https://franz.com/agraph/gruff/ (visited on Dec. 5, 2018) (cit. on p. 57).

Hayes, Patrick (2004). *RDF Semantics*. URL: https://www.w3.org/TR/rdf-mt/ (visited on Oct. 30, 2018) (cit. on pp. 27, 89).

Herman, Ivan (2008). *SPARQL is a Recommendation*. URL: https://www.w3.org/blog/SW/2008/01/ (visited on Nov. 1, 2018) (cit. on p. 32).

Litzel, Nico (2018). *Was ist Natural Language Processing?* URL: https://www.bigdata-insider.de/was-ist-natural-language-processing-a-590102/ (visited on Nov. 10, 2018) (cit. on p. 44).

Morse Documentation (2010). *Semantic camera*. URL: http://www.openrobots.org/morse/doc/1.1/user/sensors/ (visited on Dec. 5, 2018) (cit. on p. 53).

Openrobot, MORSE (2009). *The MORSE Simulator*. URL: https://www.openrobots.org/morse/doc/ (visited on Dec. 5, 2018) (cit. on p. 50).

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). *You Only Look Once: Unified, Real-Time Object Detection* (cit. on p. 16).

Robotic Industries Association (2018). *Robotic Simulation Software*. URL: https://www.robotics.org/blog-article.cfm/Robot-Simulation-Software-The-Value-for-Manufacturers/57 (visited on Aug. 5, 2018) (cit. on pp. 17, 50).

Semantic Annotation (2013). *Open Annotation Data Model*. URL: http://www.openannotation.org/spec/core/ (visited on Oct. 31, 2018) (cit. on p. 28).

Semantics RDF 1.1 (2004). *W3C Recommendation*. URL: https://www.w3.org/TR/rdf-mt/ (visited on Oct. 31, 2018) (cit. on pp. 28, 29).

Semantics, RDF 1.1 (2014). *Blank nodes*. URL: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/{\#}blank-nodes (visited on Oct. 26, 2018) (cit. on p. 25).

Tollefson, Jeff (2011). *US launches eco-network*. URL: http://www.nature.com/doifinder/10.1038/476135a (visited on Nov. 6, 2018) (cit. on p. 40).

Tully Foote (2018). *Documentation - ROS Wiki*. URL: `http://wiki.ros.org/` (visited on Dec. 5, 2018) (cit. on p. 55).

Web, Semantic (2015). *Semantic Web - W3C*. URL: `https://www.w3.org/standards/semanticweb/` (visited on May 12, 2018) (cit. on pp. 19, 20, 24).

Yaw angle calculation (2015). URL: `https://physics.stackexchange.com/questions/185204/yaw-angle-calculation-for-a-two-wheeled-inverted-pendulum` (visited on Feb. 9, 2019) (cit. on p. 69).

List of Figures

List of Tables

Abbreviations

A3ME Agent-based Middleware Approach for Mixed Mode Environments

AI artificial intelligence

AMOR Autonomous Mobile Outdoor Robot

CNN convolutional neural network

CORA Core Ontologies for Robotics and Automation

DL Description Logic

EL Entity Language

ENIAC Electronic Numerical Integrator and Computer

FCN fully convolutional network

GAT Grounded Action Transformation

GPS Global Positioning System

GSL Grounded Simulation Learning

HRI Human-Robot Interaction

HTML Hyper Text Markup Language

IBM International Business Machines

IoT Internet of Things

IR Information Retrieval

IRI Internationalized Resource Identifier

LAT Lexical Answer Type

LIS library and information science

MAS Multi Agent Systems

MIT Massachusetts Institute of Technology

MORSE Modular OpenRobots Simulation Engine

N3 Notation3

NLP natural language processing

OAQA Open Advancement of Question Answering

ORA Ontologies for Robotics and Automation

OWL Ontology Web Language

PROTEUS Plateforme pour la Robotique Organisant les Transferts Entre Utilisateurs et Scientifiques

QA Question answering

QL Query Language

R.U.R Rossumovi Univerzální Roboti

RAS Robotics and Automation Association

RDF Resource Description Framework

RDFS Resource Description Framework Schema

RDQL RDF Data Query Language

RGB Red Green and Blue

RGBA Red Green Blue Apha

RGBD Red Green Blue Depth

RL Rule Language

ROS Robot Operating System

RQL	RDF Query Language
RRI	Robot-Robot Interaction
RoboSemProc	Robot Semantic Protocol
SeRQL	Sesame RDF Query Language
SGML	Standard Generalized Markup Language
SLAM	Simultaneous Localization and Mapping
SP	Singleton Property
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
SSN	Semantic Sensor Network
SUMO	Suggested Upper Merged Ontology
SV	Selective Verb
SW	Semantic Web
UAV	Unmanned Aerial Vehicle
UIMA	Unstructured Information Management Architecture
URI	Uniform Resource Identifier
URL	Universal Resource Locator
URN	Uniform Resource Name
USAR	urban search and rescue
VOC	visual object classes
W3C	World Wide Web Consortium

WISNO	WIreless Sensor Networks Ontology
WWW	World Wide Web
XML	Extensible Markup Language
YodaQA	Yet anOther Deep Answering pipeline Question Answering
YOLO	You Only Look Once