Ontologenius : A long-term semantic memory for robotic agents

Guillaume Sarthou[1], Aurélie Clodic[1] and Rachid Alami[1]

***Abstract*— In this paper we present Ontologenius, a semantic knowledge storage and reasoning framework for autonomous robots. More than a classic ontology software to query a knowledge base and a first-order internal logic as it can be done for web-semantics, we propose with Ontologenius features adapted to a robotic use including human-robot interaction. We introduce the ability to modify the knowledge base during execution, whether through dialogue or geometric reasoning, and keep these changes even after the robot is powered off. Since Ontologenius was developed to be used by a robot which interacts with humans, we have endowed the system with ability to perform attributes and properties generalization and with the possibility to model and estimate the semantic memory of a human partner and to implement theory of mind processes.**

This paper presents the architecture and the main features of Ontologenius as well as examples of its use in robotics applications.

I. INTRODUCTION

If we envision the daily use of cognitive and interactive robots, we must be able to design a system that will be able to deal with intricate environments, to perform complex tasks, to learn through interaction and to run without stopping for hours. Moreover, we have to assume that the robot users could be a naive which calls for the ability to gather knowledge incrementally through interaction and collaborative problem solving. In addition, after stop, the robot should be able to restart with the knowledge accumulated from previous interactions. Finally, we must take into account that the on-board robot computing power is limited and that resources provided by the network which could be used to overcome this limitation can be interrupted. Consequently, the software must be designed according to these constraints.

We propose a way for the robot to acquire new knowledge during interaction and a software to store and to access this knowledge in the framework of long-term interaction [1]. In this paper, we focus on the representation of semantic knowledge by taking inspiration from long-term human memory structures coming from cognitive psychology models. We consider only the representation of the recollection of facts and general knowledge and not the recollection of previous experiences which would be the next step of our work.

The resulting software is Ontologenius, a lightweight open-source ROS-compatible software which enables to store semantic knowledge, to explore it efficiently and to reason about it. Ontologenius makes it possible to share this semantic knowledge among all the components of a robotic architecture thus enabling a uniqueness of knowledge. Additionally, Ontologenius allows to deal with several distinct knowledge bases, corresponding to the knowledge of the robot and the estimated knowledge of its human partners, while taking into account the notion of basic common knowledge. This feature enables the robot to reason about several distinct perspectives and to develop theory of mind capabilities.

We will first define the context of this work with relation to cognitive psychology and cognitive architectures. We will then discuss the choice of using an ontology before presenting the design choices and functionalities of the Ontologenius software. The last sections are dedicated to the evaluation of its performance and to illustrative examples of its use.

*The research leading to these results has received funding from the European Unions H2020 programme under grant agreement No. 688147, MuMMER `http://mummer-project.eu/`

[1]Authors are with LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France `firstname.surname@laas.fr`

II. BACKGROUND

Long-term memory (LTM) refers to memory that involves the storage and recall of information over a long period of time: "lasting days, weeks, and, in some cases, even a lifetime"[2]. There is no consensus about memory organization and several models have been proposed. Squire's model [3] divides LTM in two subparts, declarative memory (which allows a "conscious recollection about facts and events") and non-declarative memory ("is expressed through performance rather than recollection"). He also specifies declarative memory as semantic memory and episodic memory. Tulving's model [4] defines episodic memory as memory of past experienced events ("remembering") and semantic memory as the encyclopedic memory independent of the memory of the context of acquisition ("knowing").

He also defines "an SPI model of relations between these systems such that encoding into semantic and episodic systems is serial (S), storage is parallel (P), and retrieval is independent (I). Given some minimal registration of the occurrence of an event, that event may only be stored in the semantic system. Given more attention at encoding and more conscious control, the event may be further encoded into episodic memory. Events may be stored in both systems but retrieved independently from them."[5]. This theory can not be proven and must therefore be taken as [4] "an explicit starting point for a more systematic pursuit of what is clearly the next problem that needs to be tackled". Based on the SPI hypothesis, we will consider that episodic memory is dependent of semantic memory and that semantic memory is the first building block of declarative memory as shown in Figure 1.

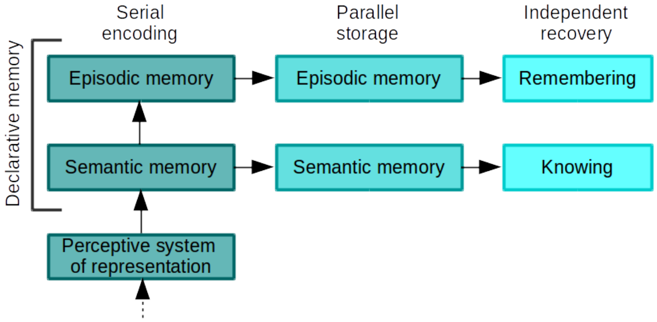

Fig. 1: Relations between episodic memory and semantic memory according to the SPI (Serial, Parallel and Independent) model of Tulving (1995). We consider that episodic memory relies upon semantic memory and that semantic memory is the first building block of declarative memory.

A. Cognitive Architectures

Cognitive Architectures (CA) have been developed to design and implement autonomous agents. The purpose can be to perform complex tasks [6], as manipulation and planning [7] [8], to reproduce human behaviors, as social emotion and creativity [9], to understand verbal and non-verbal feedback [10], etc (see [11] for a survey of the most recent ones). Knowledge representation among all these architectures as well as underlying models of memory are heterogeneous. ACT-R [12] uses a long term production memory with a declarative module; Casimir [13] only mentions a long term memory; Cerebus [14] implements a semantic net; CoJACK [15] uses declarative chunks; DSO-CA [16] uses an episodic module; EPAM [17] specifies a long term memory as a procedural and declarative memory; SOAR [18] splits the long term memory into a semantic network, a past experience memory and a procedural memory; KnowRob [7] implements a semantic network (as an ontology) and collections of episodic memory; [6] uses a symbolic facts and beliefs management as an ontology [19]. We nevertheless notice that the notions of long-term memory, declarative memory and semantic network are found in most of these architectures.

III. RATIONALE

We aim to develop a system, for robotic applications, to represent and deal with knowledge stored in what is called long-term memory (LTM) in cognitive psychology. In this paper, we will present a first step toward this goal with the implementation of what can be compared to a semantic memory. The development of a system that can be compared to an episodic memory, such as [20], would then be a second step for future work since we consider semantic memory as a building block for the declarative system.

The choice of the representation has been driven by several models. Collins and Quillian [21] proposed to model semantic memory as a hierarchical semantic network where the network nodes are the stored concepts and the links are inclusion relationships. This first model was later developed to no longer consider the semantic network purely as a hierarchical tree but just as a relation network [22]. This arrangement allows, among other things, to take into account the notion of specificity of certain examples among a category of concept, such as the fact that a kiwi is a kind of bird which cannot fly. We have selected this semantic relation network representation.

To store it, we have chosen to develop an ontology-based system. Ontology description has already been largely formalized with XML syntax and ontologies are widely used in e.g. semantic web [23]. The use of this formalism allows us among other things to directly load ontologies from Internet. This facilitates the extension of the knowledge base of the robot by sharing knowledge bases between artificial agents (as allowed by Open-EASE [24] with its sharing of procedures). In addition, research has been undertaken to standardize ontologies for robotics [25]. Although this notion of semantic network is found in several Cognitive Architectures such as Cerebus [14], Soar [18] and KnowRob [7]. Cerebus and Soar do not use ontology. KnowRob is an ontology-based system but, as far as we know, does not include the estimation of the knowledge of another agent. The software closest to our work is ORO [19] used in [6]. We bring some new features, such as generalization, reasoners as plugin modules and the management of set of distinct knowledge bases, as it will be explained in the next sections.

IV. DESIGN AND FEATURES

One design question if we envision an ontology-based system is whether to store the knowledge base remotely (and access it through web-services as it is done in the Semantic Web Domain) or locally. The use of web clients can bring many benefits such as virtually unlimited storage space and a computing power well beyond the one embedded on a robot. In addition, this would make possible to share or exchange knowledge between several robots. However, it brings also several security issues [26] and it means a total dependence on networks accessed through Internet, which can lead to slowdowns. We have made the choice to store and manage our knowledge base locally on the robot because we consider important to be able to access this knowledge anytime and to answer requests at high rate (thousand requests per second). Nevertheless, our system allows to get ontologies from the Internet and to add them to the knowledge base.

An interesting feature presented in [21] is the concept of generalization. Collins took the example of the generalization that birds can fly, which allows us to deduce that ”a canary can fly” or that other birds, although unknown, can fly. This feature can help knowledge acquisition and reduce the amount of data to be recorded. However, the addition of this feature calls for several design changes. While conventional ontologies only apply properties to individuals, we need to be able to apply properties to classes to represent generalization (i.e. we need to be able to encode that ”a bird can fly” and not only that ”a canary can fly”). Then, we need also to take into account that a property that applies to a class, may be wrong for one individual, even if it is part of the class (e.g. a kiwi is a bird but cannot fly). Finally, if we enable the system to deduce and generalize facts, e.g. by presenting several individuals with the same property the system will deduce that this property applies to the class, we must have in mind that this can be wrong. We have chosen to make a distinction between the knowledge that has been deduced and the knowledge that has been acquired by other means. This opens the possibility for the robot to question some part of its knowledge whenever it is possible (e.g. by asking its human partner if something that has been generalized is true or not).

Another feature needed in a human-robot interaction is theory of mind management. Theory of mind is the ability "to conceive of mental states: that is, knowing that other people know, want, feel, or believe things" [27]. To do so, our system would need to represent and reason about the (semantic) knowledge of the other agents. ORO software [19] already offered this functionality but stored the knowledge in a unique ontology. We have made the choice to store the estimation (by the robot) of each agent knowledge in a dedicated ontology file (one ontology file per agent), which enables to load it again for future interaction. This choice refers to the concept of "self-other distinction" according to which "for shared representations (...) to foster coordination rather than create confusion, it is important the agents be able to keep apart representations of their own and other's actions and intentions" [28]. In addition, we offer the possibility to load at first a basic knowledge base, which could represent common ground.

V. ARCHITECTURE

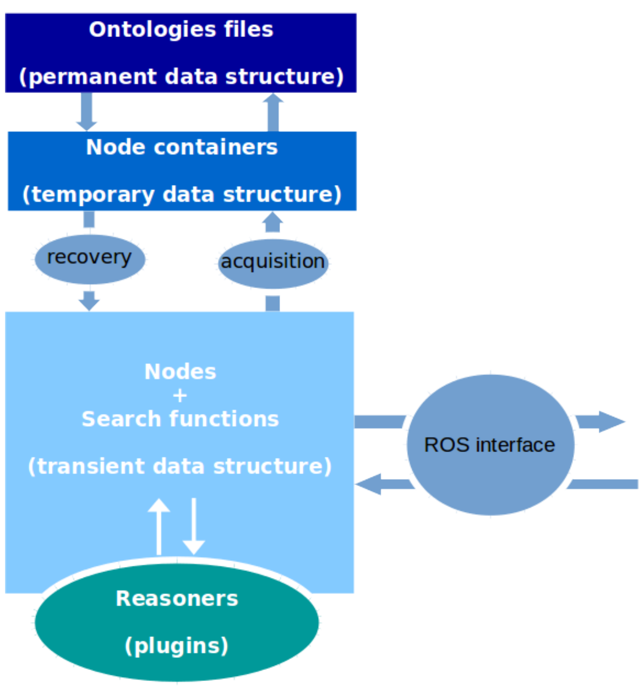

Fig. 2: Ontologenius Software Architecture.

The Ontologenius architecture is divided into three major modules as shown in Figure 2: the long-term storage module (permanent and temporary data), the knowledge base exploration module (transient data) and the reasoning module (plugins). Ontologenius core and API have been developed and is available in C++14. In addition, a full ROS interface is available (with support from ROS Kinetic to Melodic).

A. Semantic nodes

A semantic node is the lower data structure in Ontologenius. Each concept of the knowledge base is represented with a semantic node. We distinguish four types of nodes: individuals, classes, data properties and object properties. Each node has a set of pointers to the concepts with which it is linked. For example, the "canary" concept has a pointer to the "bird" concept among its list of inherited concepts. The set of concepts is then seen as a semantic network.

A new relation to a node can have three different labels:

- $Steady$: A steady relation is a relation inserted in the knowledge base by another component of the robotic architecture. It is a relation that can not be removed or altered by an internal Ontologenius process.
- $Inferred$: An inferred relation has been inferred from steady relationships and/or inferred relations through first-order logic reasoning. It is a relation which cannot be removed or altered by an internal Ontologenius estimation process. The inferred relations are not stored in the permanent data structure when Ontologenius is powered down, as they can be inferred again. This allows a storage gain.
- $Estimated$: Unlike the inferred relations which are necessarily true with respect to the rest of the knowledge base, the estimated relations come from reasoning not relying on first order logic. Such relationships can then be false. This is the case of relations resulting from the mechanism of generalization. Estimated relations are the only relation which can be removed or altered by an internal Ontologenius process.

A semantic node is defined by a unique identifier in the form of a string of characters and an other unique identifier in the form of an integer. The first one allow to find a node using the name of the concept will the second allow a fast comparison of nodes. In addition, a dictionary is available for each node to link this identifier to its natural language expression. To do so, the dictionary links several strings of characters to a given identifier (and dedicated concept). For example, the concept "cup" can have the identifier "obj24_cup" and have the dictionary "en:"cup, goblet", fr:"tasse, gobelet", es:"taza"". At launch or during execution, a working language can be chosen. In that case, the search for an identifier can take advantage of the dictionary. For example, working in french and looking for a "tasse", Ontologeniuswill find "tasse" in the french dictionary of the node "obj24_cup" and will send back this identifier. This choice has a positive impact on performance which will be presented in section VI. It should be noted that a same verbalization might be used for several concepts, it is quite admitted and even makes it possible to highlight a possible ambiguity that can be removed thanks to other semantic characteristics of similar concepts.

B. Long-term storage

Long-term storage falls into two storage strategies depending on whether Ontologenius is running or not. Out of operation, it is realized using Ontologies OWL files with XML syntax. These files can be stored locally on the robot or on a remote server. At Ontologenius startup, a list of ontology files specified by the user are loaded. When loaded, the concepts described in the files are converted into nodes and organized into nodes containers. The nodes containers are composed of a map for which the keys are the identifiers of the nodes and the values are the nodes themselves. The use of a map provides the complexity of finding an identifier in a container in log2 (n). The knowledge base exploration algorithms can thus perform a recovery of the work nodes from their identifiers and propagate in the semantic network through the relations links.

C. Reasoning modules

Ontologenius provides reasoning algorithms based on first-order logic. Among them, we find the resolution of symmetric properties, chained properties or inverse properties. Thanks to a plugin mechanism, the user can control which reasoners are used for his applications during execution, removing or adding new ones. As it will be illustrated in section VII-B, we have used this mechanism to implement the data property generalization and to enrich the dictionaries of the nodes by proposing different syntaxes for the same words in natural language.

VI. COMPUTATIONAL PERFORMANCE

Since Ontologenius was developed for embedded robotic applications, the performance of our software has been an important evaluation criterion. In addition to the measurement of CPU time under nominal conditions, we look at the evolution of CPU times in the case of large-scale knowledge bases (large-scale ontologies). All measures were done using Ontologenius as a server and creating clients using the C++ API. This means that the measurements presented include also interprocess communication times due to the use of ROS, representing its effective use in a robot software architecture. All the results presented in this section were obtained with Ubuntu 16.04 and ROS Kinetic with an Intel Core i7-7700 processor at 3.60 GHz.

For each test case and for both software, the knowledge base was first emptied and then completed by inserting the concepts one by one using the provided API. These steps were repeated for each load level. This means that the inserted concepts have no human meaning but emulate a load of knowledge at the software level.

As presented in the architecture section, a concept is identified by a unique identifier but may also have several verbalizable strings of characters. These words in natural language make it possible to interface between the internal representation of the robot and that of the human with which

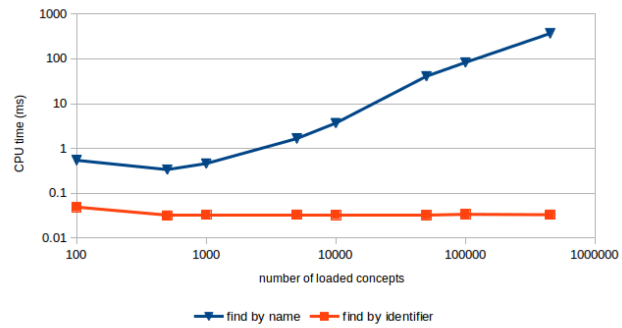

Fig. 3: Comparison of the recovery time of a concept by name and identifier.

it interacts. This need for understanding words in natural language is necessary only infrequently with respect to the frequency of manipulation of the internal concepts of the robot. As a result, the recovery time of a concept by its identifier must be as short as possible while the recovery time of a concept by one of its names in natural language can cope with an additional delay. Figure 3 indicates the average recovery time of a concept by name and identifier based on the number of concepts loaded in the knowledge base of the robot. While the recovery time increases with the number of concepts loaded, the recovery time by identifier remains somewhat stable around 0.04 ms per recovery and that even with 450,000 concepts in the knowledge base. These results were obtained by pre-inserting the N concepts into the ontology and then searching each of them by name and identifier three times each. The mean times presented here are therefore an average of 3N queries. The fact to query each of the N concept each time makes it possible to be sure to get away from the order of the concepts in the internal data structures and thus have an average value regardless of their position. Although the shapes of the curves obtained are of course consistent with the search methods used (search in a map and in a list), we can still compare here the orders of magnitudes of each of the two methods.

As mentioned above, the ORO software is the one that comes closest to our work regarding the objectives and has been a source of inspiration for our work. To evaluate the performance of Ontologenius during queries on the knowledge base we have therefore taken the evaluations proposed in [19] as well as the corresponding ontologies. Figures 4, 5 and 6 present the results for ORO and Ontologenius software for three types of requests of different complexity and depending on the number of concepts present in the knowledge base of the robot. All the tests for both software were done on the same machine and with the latest versions of the software available online. Both software were used as servers and each of the test clients use the provided C++ APIs.

It is important to note that ORO was not developed with the objective of being efficient in performance which explains the important differences that we will present. Nevertheless, it is the software with the paradigm closest to ours and therefore the most intersting with which we can compare ourselves.

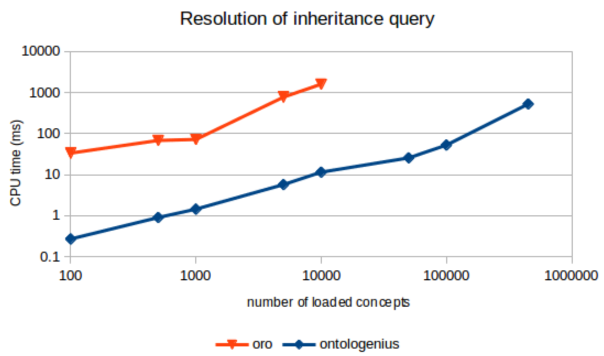

Fig. 4: CPU time of inheritance query according to the number of individuals described in the ontology for ORO and Ontologenius.

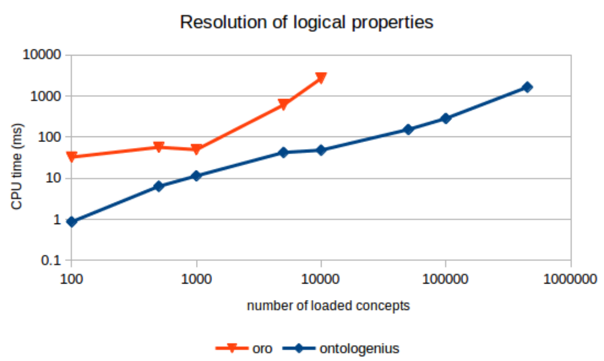

Fig. 5: CPU time of logical properties query according to the number of individuals described in the ontology for ORO and Ontologenius.

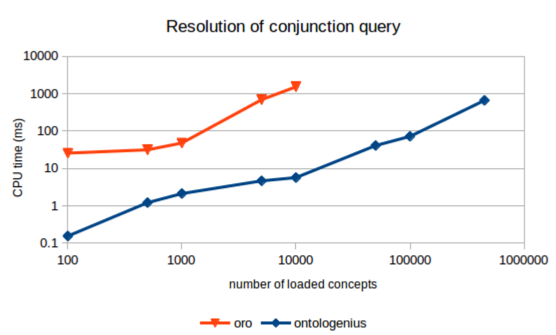

Fig. 6: CPU time of conjunction query according to the number of individuals described in the ontology for ORO and Ontologenius.

These results show a better performance for Ontologenius ranging from a factor of 4.33 for the resolution of logical properties to a factor of 267.7 for the resolution of conjunction queries. We also note that the ORO software could not be tested beyond 10,000 concepts loads given the rise of an error beyond. In spite of a progressive increase of the CPU time in function of the number of concepts loads, Ontologenius has continued to function even with 450,000 concepts loads. For the moment, we have not yet determined the limit from which Ontologenius is no longer usable.

The results presented here therefore demonstrate the usability of Ontologenius for online applications, even with large-scale knowledge bases.

VII. APPLICATIONS EXAMPLES

We briefly describe herebelow four test-cases. These four scenarios are available in a simplified version (i.e. no robot needed) in tutorials available on the Ontologenius website[1].

A. *Who is the intruder?*

In this game, three cards, each illustrating a concept, are presented to a player (e.g. a horse, a dog and a robot). The goal is to find the intruder (e.g. the robot) and to give an explanation (e.g. a robot is not an animal).

In our implementation, we present to the robot at least three cards, each one illustrating a concept (an individual from an ontology). From its knowledge base [2] the robot must find the concepts having the least relation with the others and explain to the human the reasons of its choice. This example illustrates the search for information through the inheritance contained in the knowledge base. The interest of this scenario is to show that it is possible to look for the most pertinent answer to give to the human partner. This scenario was implemented with the Pepper robot and the use of an AR tag perception module[3]. The identifiers of the tags are mapped with the identifiers of the concepts. The response of the robot is done verbally with the Pepper text to speech module.

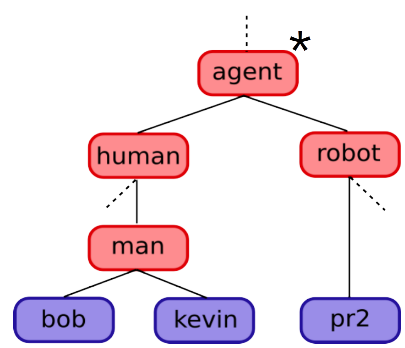

Fig. 7: Inheritance relations of the concepts ”Bob”, ”Kevin” and ”Pr2”. The class ”agent” (*) represents the common point between the three concepts.

In the video, we show the robot the concepts ”Bob”, ”Kevin” and ”Pr2”. As represented in Figure 7, Bob and Kevin are described as being men. A man is described as being a human and a human being as a type of agent. PR2

[1]https://sarthou.github.io/ontologenius

[2]https://github.com/sarthou/ontologenius/blob/master/ files/tutorials/tutorial_1.owl

[3]https://www.youtube.com/watch?v=JHnTmE_ib-M

is described as being a robot and a robot is also a type of agent (being the common point between the three concepts). The robot then responds that the intruder is Pr2 given that the others are humans. Through the exploration of the knowledge of the robot we have been able to determine that it is more relevant to notify that Pr2 is not a human rather than to notify that he is not a man.

B. *Do the birds fly?*

This scenario corresponds to the example of [21] to present the concept of generalization. In this scenario, we first introduce to the robot three birds that can not fly. We then introduce a new bird, bird B, with no information on its ability to fly and ask the robot if this new bird could fly. Due to the generalization process, Ontologenius estimates that birds (the class) can not fly, so the robot responds that bird B can not fly.

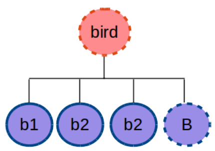

Fig. 8: The three birds b1, b2 and b3 are described as unable to fly (circles). The concept ”bird” and bird B are then considered (- - -) as not being able to fly.

In a second time, we present to the robot eight new birds able to fly and ask again to the robot if the bird B can fly. This time, the robot responds that bird B can fly. As Ontologenius knows only three birds that can not fly and eight that can do so, it has changed its estimates of the ability of birds to fly.

Here the threshold of generalization is of 60% but this is configurable in the plugin of generalization. This explains why we have to add eight flying birds after teaching three who can not fly.

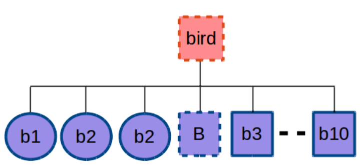

Fig. 9: The eight birds b3 to b10 are described as able to fly (square boxes). The concept ”bird” and ”bird B” are then now considered (- - -) as being able to fly.

If the robot is switched off and on again the twelve birds will still be known to the robot and bird B will still be considered capable of flying. Although the robot has deduced that the general bird concept can fly, birds 1 to 3 are always considered as not being able to fly. These are special cases. This scenario was implemented with an Amazon Echo™with Alexa to add new knowledge to Ontologenius and query it verbally. It can be used to extend the examples presented with the human-robot interactive learning architecture of Angleraud et al. [29] in which a human teaches a robot that spaghetti are pasta so that the robot infers how to cook them knowing how to cook pasta.

C. *Find your way*

To describe the topography of a place such as a shopping center, a topological graph is the most appropriate representation. However, when we want to add semantic information such as the types of stores or the items they sell, the use of an ontology becomes relevant. To avoid having several representations for the same environment according to the algorithm used, we illustrate with Ontologenius that it is possible to represent the topology of an environment as well as the associated semantic information in an ontology [30]. In addition to topological and semantic information, we have shown that it is possible to represent the geometric relationships between elements of the environment. These geometric relationships allow us to provide all the information necessary to generate a route description speech using a route perspective and referring to landmarks based on the estimated future position of the guided human. A basic description of a place would look like follow:

```
<owl:NamedIndividual rdf:about=”Burger_King”>
  <rdf:type rdf:resource=”restaurant”/>
  <X:isInFrontOf rdf:resource=”Espresso_House”/>
  <X:isAlong rdf:resource=”Keskuspuisto”/>
  <X:hasAtRight rdf:resource=”Rax”/>
  <rdfs:label xml:lang=”en”>Burger King
   </rdfs:label>
</owl:NamedIndividual>
```

Ontologenius-based path search and verbalization algorithms have been tested in a shopping center [31], [32] that has been described in an ontology of more than 550 concepts (see Figure 10). In this ontology, we have represented the stores, their types, the items they sell and the geometric relationships between stores. The search algorithm was able to determine 20 different routes from point A to point B by analyzing 129 paths in less than 200 ms. Thanks to the performances provided by Ontologenius, although the representation is not directly dedicated to search algorithms we noticed that the CPU times using a purely semantic representation of an environment are quite acceptable for a human-robot interaction.

D. *Providing basic mechanisms for Sally&Anne test*

Sally & Anne test [27] is used in developmental psychology to assess, in social cognition, the ability of a child to understand that someone else has mental states different from his own. Our robot control architecture has been designed to integrate the associated observation and situation assessment abilities [33], [34], [35]. The symbolic level reasoning can be implemented using Ontologenius. To do so, we create an Ontologenius instance to represent Sally's semantic knowledge, one for Anne's and one for the observer (the robot) as shown on Figure 11. First of all, the three knowledge bases are loaded with a common ground that is at least knowledge of the existence of both containers, the ball and some geometric properties like ”is in”. When Sally puts the ball in the first container while Anne is present at time $t1$,

Fig. 10: Robot describing a route to a human in a mall using algorithms based on Ontologenius. The sentence in green is the explanation of the route verbalized by the robot and generated using the semantic representation stored in Ontologenius: "just go down the corridor and then go almost at the very end of this corridor and it's on the left when you walk".

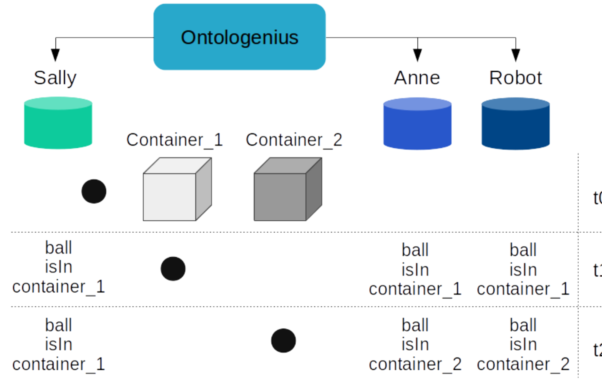

Fig. 11: Insertions of different knowledge about the position of the ball between the knowledge bases of several agents.

"ball isIn container_1" is put into in Sally's knowledge base as well as Anne's and the robot's knowledge base. When Anne moves the ball into the second container while Sally is away at time $t2$, only Anne's and the robot's knowledge are updated with "ball isIn container_2".

Ontologenius provides specific queries. For example, it is possible to ask Ontologenius about the difference of knowledge regarding "ball" between Sally and the robot. Using the Ontologenius API, we typically query a specific knowledge base by accessing an instance by name (e.g. kb["sally"]). To ask questions about a difference in knowledge, we can use the API at a higher level with the $getDifference$ function (ie getDifference("sally", "robot", "ball")). In that case, Ontologenius will return: [+]ball|isIn|container_1, [-]ball|isIn|container_2. This can be read as the fact that "ball isIn container_1" is present in the Sally's base and not in one of the robot and that the fact that "ball isIn container_2" is not present in Sally's base while it is in the robot's. The difference of knowledge between Anne and the robot about the ball gives an empty result which means that there is no difference of knowledge. Ontologenius helps to detect divergence of beliefs between agents and so could be used in application where theory of mind management is needed. However, it is impossible for Ontologenius to affirm which of the knowledge bases contains the field truth. This must come from a higher level interpretation.

VIII. CONCLUSIONS

In this paper, we have presented Ontologenius, a semantic knowledge processing software for robotic applications. Ontologenius is a server based on ROS for an easy deployment on various robotic platforms. It maintains several knowledge bases, one for each agent interacting with the robot while running background processes such reasoning and knowledge manipulation processes independently for each knowledge base. Ontologenius allows the addition of external ontologies from the Internet and the long-term safeguarding of the robot accumulated knowledge. Since Ontologenius defines names that can be verbalized for each concept, it is suitable for verbal interactions either to add knowledge or to query the ontology. With the ability to apply properties to ontology classes, Ontologenius offers the possibility to perform reasoning, such as generalization, which allows to estimate new knowledge.

Ontologenius' evaluation (regarding both temporal performance and the amount of data it can store and manipulate) demonstrates its on-line usability in semantically rich environments.

Even though Ontologenius is aimed to be low level to guarantee high performance, the ontologenius_query[4] ROS package allows to query the knowledge base using SPARQL queries, though this may degrade its performance.

In summary we proposed and discussed the main features of a module that can be assimilated to a semantic memory for robots as a basic component of its declarative memory. From this basis, we will now consider the development of a software dedicated to the storage and manipulation of the episodic knowledge of a cognitive and interactive robot. In this future work, the memory aspect over the long-term should be at the center of the software architecture to take into account both the storage space limit of a robot that the activation and the recovery time of a distant memory.

REFERENCES

[1] I. Leite, C. Martinho, and A. Paiva, "Social Robots for Long-Term Interaction: A Survey," *International Journal of Social Robotics*, vol. 5, no. 2, pp. 291–308, Apr. 2013.

[2] C. Bailey, D. Bartsch, and E. Kandel, "Toward a molecular definition of long-term memorystorage," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 24, pp. 13 445–13 452, Nov. 1996. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC33629/

[3] L. R. Squire, "Memory systems of the brain: A brief history and current perspective," *Neurobiology of Learning and Memory*, vol. 82, no. 3, pp. 171–177, Nov. 2004.

[4] E. Tulving, "Organization of memory: Quo vadis," *The cognitive neurosciences*, vol. 839847, 1995.

[5] J. H. Byrne, *Learning and memory: a comprehensive reference*. Elsevier., 2008.

[4]https://sarthou.github.io/ontologenius_query

[6] S. Lemaignan, M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami, “Artificial cognition for social humanrobot interaction: An implementation,” *Artificial Intelligence*, vol. 247, pp. 45–69, June 2017.

[7] M. Beetz, D. Beler, *et al.*, “Know Rob 2.0 A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 512–519.

[8] D. Kraft, E. Baeski, *et al.*, “Exploration and Planning in a Three-Level Cognitive Architecture,” *Proceedings of the International Conference on Cognitive Systems (CogSys 2008)*, pp. 71–78, Apr. 2008.

[9] A. V. Samsonovich, “Intellectual Agents Based on a Cognitive Architecture Supporting Humanlike Social Emotionality and Creativity,” in *Advances in Neural Computation, Machine Learning, and Cognitive Research II*, B. Kryzhanovsky, W. Dunin-Barkowski, V. Redko, and Y. Tiumentsev, Eds. Cham: Springer International Publishing, 2019, pp. 39–50.

[10] D. E. Cahya and M. Giuliani, “Towards a cognitive architecture incorporating human feedback for interactive collaborative robots,” *Lecture Notes in Computer Science*, vol. 10965, pp. 486–488, July 2018.

[11] P. Ye, T. Wang, and F. Wang, “A Survey of Cognitive Architectures in the Past 20 Years,” *IEEE Transactions on Cybernetics*, vol. 48, no. 12, pp. 3280–3290, Dec. 2018.

[12] J. R. Anderson, C. J. Lebiere, and C. J. Lebiere, *The Atomic Components of Thought*. Psychology Press, Jan. 2014.

[13] H. Schultheis and T. Barkowsky, “Casimir: An Architecture for Mental Spatial Knowledge Processing,” *Topics in Cognitive Science*, vol. 3, no. 4, pp. 778–795, 2011.

[14] I. Horswill, “Cerebus: A Higher-Order Behavior-Based System,” *AI Magazine*, 2002.

[15] R. Evertsz, M. Pedrotti, P. Busetta, H. Acar, and F. E. Ritter, “Populating VBS2 with Realistic Virtual Actors,” p. 8, 2009.

[16] X. Pan, L. N. Teow, K. H. Tan, J. H. B. Ang, and G. W. Ng, “A cognitive system for adaptive decision making,” in *2012 15th International Conference on Information Fusion*, July 2012, pp. 1323–1329.

[17] H. B. Richman and H. A. Simon, “Simulations of Classification Learning using EPAM VI,” p. 64, 2002.

[18] J. E. Laird, A. Newell, and P. S. Rosenbloom, “SOAR: An architecture for general intelligence,” *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, Sept. 1987.

[19] S. Lemaignan, R. Ros, L. Msenlechner, R. Alami, and M. Beetz, “ORO, a knowledge management platform for cognitive architectures in robotics,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 3548–3553.

[20] A.-L. Mealier, G. Pointeau, *et al.*, “Narrative Constructions for the Organization of Self Experience: Proof of Concept via Embodied Robotics,” *Frontiers in Psychology*, vol. 8, 2017.

[21] A. M. Collins and M. R. Quillian, “Retrieval time from semantic memory,” *Journal of Verbal Learning and Verbal Behavior*, vol. 8, no. 2, pp. 240–247, Apr. 1969.

[22] A. M. Collins and E. F. Loftus, “A spreading-activation theory of semantic processing,” *Psychological Review*, vol. 82, no. 6, pp. 407–428, 1975.

[23] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[24] M. Beetz, M. Tenorth, and J. Winkler, “Open-EASE,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1983–1990.

[25] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Miguelez, “An IEEE standard Ontology for Robotics and Automation,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 1337–1342.

[26] T. Denning, C. Matuszek, K. Koscher, J. R. Smith, and T. Kohno, “A Spotlight on Security and Privacy Risks with Future Household Robots: Attacks and Lessons,” in *Proceedings of the 11th International Conference on Ubiquitous Computing*, ser. UbiComp ’09. New York, NY, USA: ACM, 2009, pp. 105–114, event-place: Orlando, Florida, USA.

[27] S. Baron-Cohen, A. M. Leslie, and U. Frith, “Does the autistic child have a theory of mind ?” *Cognition*, vol. 21, no. 1, pp. 37–46, Oct. 1985.

[28] E. Pacherie, “The phenomenology of joint action: Self-agency vs. joint-agency,” in *Joint Attention: New Developments*, S. Axel, Ed. MIT Press, 2012.

[29] A. Angleraud, Q. Houbre, V. Kyrki, and R. Pieters, “Human-Robot Interactive Learning Architecture using Ontologies and Symbol Manipulation,” in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug. 2018, pp. 384–389.

[30] G. Sarthou, R. Alami, and A. Clodic, “Semantic Spatial Representation: a unique representation of an environment based on an ontology for robotic applications,” in *SpLU-RoboNLP 2019*, Minneapolis, United States, June 2019, pp. 50 – 60. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02177140

[31] M. E. Foster, R. Alami, O. Gestranius, O. Lemon, M. Niemelä, J.-M. Odobez, and A. K. Pandey, “The MuMMER Project: Engaging Human-Robot Interaction in Real-World Public Spaces,” in *Social Robotics. 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016 Proceedings*, ser. Lecture Notes in Computer Science, Oct. 2016, pp. 753–763.

[32] J. Waldhart, A. Clodic, and R. Alami, “Reasoning on Shared Visual Perspective to Improve Route Directions,” in *IEEE RO-MAN 2019*, Edinburgh, United Kingdom, Aug. 2014, p. FrAT2.4.

[33] E. A. Sisbot, R. Ros, and R. Alami, “Situation assessment for human-robot interactive object manipulation,” in *20th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2011, Atlanta, Georgia, USA, July 31 - August 3, 2011*, 2011, pp. 15–20.

[34] G. Milliez, M. Warnier, A. Clodic, and R. Alami, “A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management,” in *IEEE RO-MAN 2014*, Edinburgh, United Kingdom, Aug. 2014, p. FrAT2.4.

[35] S. Lemaignan, Y. Sallami, C. Wallbridge, A. Clodic, T. Belpaeme, and R. Alami, “UNDERWORLDS: Cascading Situation Assessment for Robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain, Oct. 2018. [Online]. Available: https://hal.laas.fr/hal-01943917