Integrating Ontologies with Multi-Agent Systems through CArtAgO Artifacts

Artur Freitas, Alison R. Panisson, Lucas Hilgert,
Felipe Meneguzzi, Renata Vieira, Rafael H. Bordini
Postgraduate Programme in Computer Science, School of Informatics (FACIN)
Pontifical Catholic University of Rio Grande do Sul (PUCRS). Porto Alegre, RS - Brazil
{artur.freitas, alison.panisson}@acad.pucrs.br,
{lucas.hilgert, felipe.meneguzzi, renata.vieira, rafael.bordini}@pucrs.br

***Abstract*—Several advantages can be obtained by allowing multi-agent systems to easily access ontologies, for example, in scenarios where agents make their decisions based on the knowledge provided by ontologies. Thus, this paper presents an infrastructure to allow the use of web ontologies in different agent-oriented platforms. The agents use this infrastructure layer as a tool for storing, accessing and querying domain-specific OWL ontologies. As a result, this layer allows an integration of agent platforms with semantic web data and ontologies. We exemplify in practice how agents, coded in one such platform, can use the proposed access layer to ontological reasoning engines, as well as which features can be obtained from it. The performance of this semantic infrastructure is evaluated and compared against usual knowledge representation in agent programming.**

***Keywords*-Ontology, Agent-Oriented Programming Languages, Multi-Agent Systems, CArtAgO**

I. Introduction

Ontologies empower the execution of semantic reasoners, such as Pellet [1], which provide the functionalities of *consistency checking*, *concept satisfiability*, *classification* and *realisation*. Ontologies also allow sharing a common understanding of the structure of information among people and software agents and the reuse of domain knowledge. The integration of such semantic technologies into Multi-Agent Systems (MAS) enhances the knowledge representation features and reasoning capabilities of applications developed under these paradigms. Using ontologies in MAS results in the possibility of creating logic rules that can be applied by a semantic reasoner to infer new knowledge. Thus, the logic is moved from the agent code to the ontology, and the knowledge may be reused by different applications. Moreover, each agent is allowed to include these ontologies and specialise them with more specific and domain-dependent knowledge.

Our approach enables the use of ontologies within MAS, by enabling agents to reason about and query elements encoded in ontologies, such as instances, concepts and properties. Agents in such systems interact with ontologies by means of an infrastructure layer coded in a CArtAgO [2] artifact (CArtAgO offers computational abstractions and provides services that agents can exploit to support their activities). The information obtained from operations over this infrastructure may be used in agent plans to achieve goals, such as in argumentation-based negotiation/dialogue scenarios, whereupon more information can benefit the agents engaged in such process [3]. Agents can use the operations of our artifact to access and manipulate information in ontologies, as we show in further sections, using the Jason [4] agent platform to access ontologies in OWL [5].

This paper makes the following contributions: *(i).* developing an infrastructure layer (artifact) coded in CArtAgO to enable ontology reasoning and querying features in different agent-oriented platforms; *(ii).* describing and implementing scenarios in Jason agent platform using the operations provided by such infrastructure; *(iii).* evaluating and comparing the performance of this new knowledge representation approach (of accessing ontologies by the infrastructure) against representations that use the belief base of agents; *(iv).* discussing advantages, limitations and trends of enabling agents to access the knowledge from ontology to support their decision making.

This paper is structured as follows. It first explains a theoretical background on multi-agent systems and ontologies. Then we propose an architecture, based on a CArtAgO artifact, working as an infrastructure layer to provide ontology manipulation capabilities in agent platforms. The following section uses this artifact to access an OWL ontology in the context of Jason agents. We explain the ontology used, reasoning examples and how it can support the decision making of agents. Next, experiments are used to compare the performance of our approach against an agent reasoning that uses only the regular agent's belief base. Finally, we discuss related work and outline research directions.

II. Ontologies and Multi-Agent Systems

Ontology is defined as an *explicit specification of a conceptualisation* [6], where a *conceptualisation* 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 conceptualisation, explicitly or implicitly [6]. Some essential properties of ontologies are [7]: *(i)* ontologies describe a specific domain; *(ii)* ontology users agree to use the terms consistently; *(iii)* ontology concepts and relations are unambiguously defined in

a formal language by axioms and definitions; *(iv)* relationships between ontology concepts determine the ontology structure; and *(v)* ontologies can be understood by computers. More importantly, ontologies empower the execution of semantic reasoners which provide functionalities such as *consistency checking*, *concept satisfiability*, *classification* and *realisation*.

Ontologies are knowledge representation structures, usually based on Description Logics, composed of concepts, properties, individuals, relationships and axioms [8]. A concept (or class) is a collection of objects that share specific restrictions, similarities or common properties; a property expresses relationships between concepts; an individual (instance, object, or fact) is an element of a concept; a relationship instantiates a property to relate two individuals; and an axiom (or rule) imposes constraints on values of concepts or individuals normally using logic languages (that can be used to check ontological consistency or infer new knowledge). The most prominent ontology language is OWL (Web Ontology Language), which is a language for processing web information and semantic web standard formalism to explicitly represent the meaning and relationships of terms [5].

We consider that these essential properties of ontologies have a role to play in multi-agent systems. Agents are reactive systems that can independently determine how to best achieve their goals and perform their tasks while demonstrating properties such as autonomy, reactivity, pro-activeness and social ability [4]. Although the advantages of ontologies for agents are clear, few multi-agent system platforms currently integrate ontology techniques. Limited ontological support is provided by agent-oriented software engineering approaches since they do not incorporate ontologies throughout the entire systems development life cycle nor consider ways in which ontologies can be used to account for interoperability and verification during design [9]. Considering such context, this work investigates the use of ontologies in multi-agent systems.

III. Engineering Ontology-based Agents

There are many agent-oriented programming platforms, such as Jason, Jadex, Jack, AgentFactory, 2APL, GOAL, Golog, and MetateM, as pointed out in [10]. Those languages differ in the agent architecture used, in the form of communication/interaction between them, and also on the programming paradigms that inspired or underlie each language. Our proposal to interact with ontologies can be used in any agent platform that supports CArtAgO. In this paper we used Jason [4] to demonstrate the application of our artifact. Jason is one of the best-known languages inspired by the BDI (*Beliefs-Desires-Intentions*) architecture. Jason is open source interpreter and offers several features such as speech-act based agent communication, plans annotation, architecture customisation, distributed execution and extensibility through internal actions.

As previously explained, ontology is defined in computer science as an *explicit specification of a conceptualisation*. In other words, it means an abstract model of some world aspect that specifies properties of important concepts and relationships. Ontologies are knowledge representation structures composed of concepts, properties, individuals, relationships and axioms. For example, OWL (Web Ontology Language) is a language for processing web information defined as a semantic web standard formalism to explicitly represent the meaning of terms and the relationships between those terms [5]. The use of ontology in agents is motivated by the needs of improving knowledge representation and enabling the execution of semantic reasoning. For example, in OWL, a given class C can be declared with certain conditions (i.e., every instance of C has to satisfy these restrictions, and/or every instance that satisfies these restrictions can be inferred as belonging to C). OWL class restrictions [5] can be defined by elements such as cardinality and logic restrictions (e.g., union, intersection, negation, the universal and the existential quantifier). These restrictions allow to make inferences by using semantic reasoners over the ontology, which are important features to provide to agent when building complex artificial intelligence systems.

A comparison about the integration of ontologies within MAS is discussed with more detail in the Related Work section. In short, AgentSpeak-DL [11] is a language which appears in a paper that does not implement it in any agent platform; JASDL [12] is an AgentSpeak-DL implementation directly in Jason; and CooL-AgentSpeak [13] is implemented in a way that each agent ontology is private. Our approach differs in the sense that ontologies can be shared among more than one agent and the ontologies can be used in several agent platforms. These features are obtained based on the architecture we designed that is implemented in CArtAgO [2]. CArtAgO is a platform to support the artifact notion in MAS. Artifacts are function-oriented computational abstractions which provide services that agents can exploit to support their activities. As design and implementation decision, each instance of our artifact can load and encapsulate exactly one OWL ontology. However, each workspace can have any number of instances of this artifact, where each instance references an ontology, and the agents in the same workspace of the artifacts can observe and manipulate any number of them. Thus, MAS using our approach can handle multiple ontologies.

The approach proposed in this paper is an alternative to agents in which the knowledge is represented and manipulated by means of ontologies, instead of using a platform-specific mechanism (such as a belief base). However, an agent may still use its regular knowledge representation approach simultaneous with the new approach proposed here (or completely replace the old approach). As this paper demonstrates, using our approach provides advantages in terms of expressiveness, interoperability, and performance. Our infrastructure layer implemented in CArtAgO provides ontology features to agents by using the OWL API [14], which allows to create, manipulate and serialise OWL ontologies. An artifact makes its functionalities available and exploitable by agents through a set of operations and a set of observable properties [2].

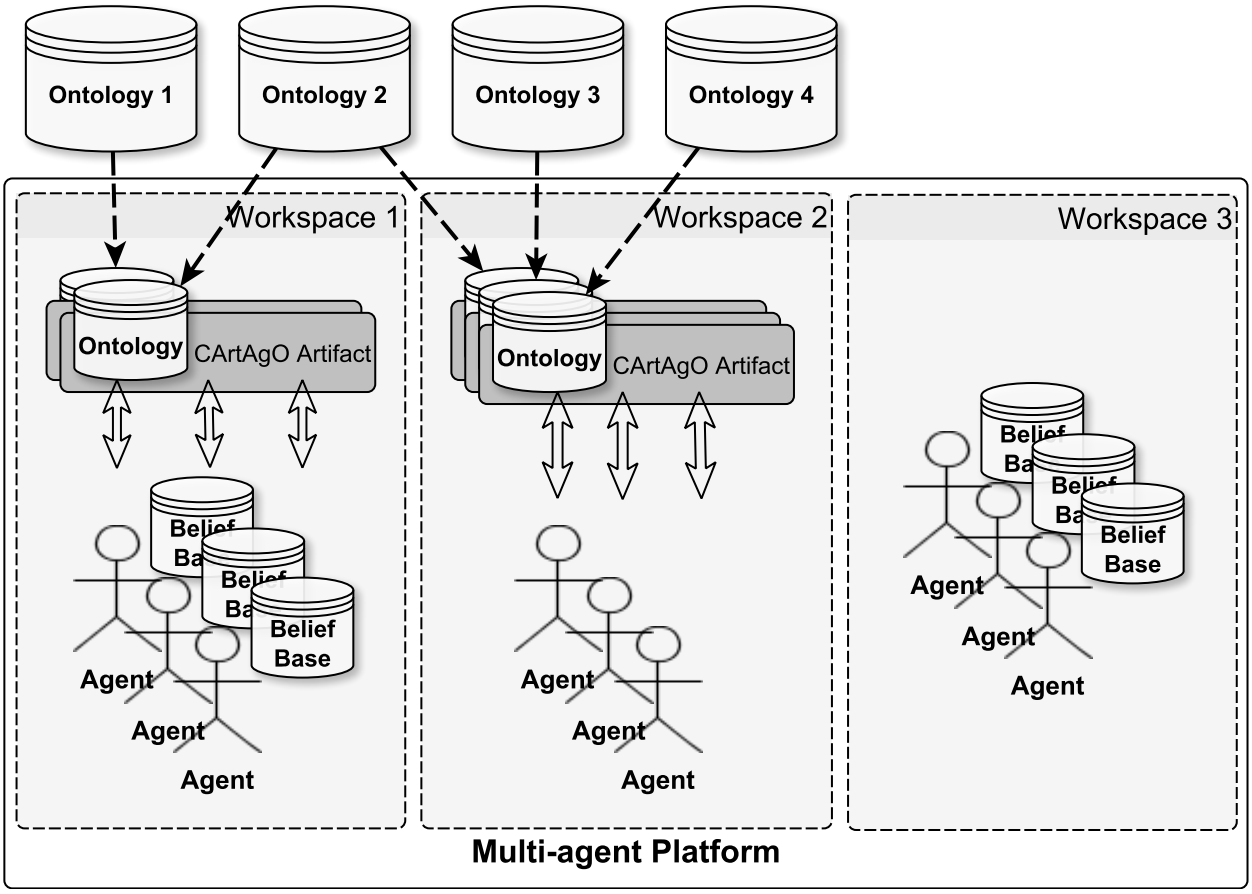

Fig. 1. Example of agents using the proposed approach (Workspace 1 and 2) versus usual multi-agent systems (Workspace 3).

Operations represent computational processes executed inside artifacts, that may be triggered by agents or other artifacts.

An example of our approach is showed in Figure 1, where we have 3 workspaces with different configurations. As described, each workspace can have any number of instances of CArtAgO artifacts, and each artifact loads and encapsulates an OWL ontology. The agents can observe and manipulate the correspondent ontologies depending on artifacts available to them in the workspace. Further, as described, the agents can still use their regular knowledge representation approach (e.g., belief base) simultaneous with the new approach proposed here, e.g., Workspace 1 in Figure 1, or completely replace the old approach, e.g., Workspace 2 in Figure 1. The usual approach, without using our proposed artifact to interact with ontologies, is shown in Workspace 3 of Figure 1.

Figure 1 clearly demonstrates that our approach allows agents to share the same ontology, including agents from different workspaces (e.g., the agents on Workspace 1 and 2 are sharing the Ontology 2), as well as it allows the agents to use information from specific ontologies, based on their role in the multi-agent system. Figure 1 shows just one possible multi-agent system configuration, we emphasise that different configurations are possible, depending on resources provided by the multi-agent platform. In other words, our approach requires that the platform used supports CArtAgO artifacts.

Our artifact provides the following operations:

- **addInstance**(instance) : adds the new instance in the ontology;
- **isInstanceOf**(instance, concept) : verifies if the instance belongs to the given concept, returning a boolean value;
- **getInstances**(concept) : retrieves a set of instances classified in a specific concept, returning a Set<OWLNamedIndividual>;
- **addProperty**(domain, property, range) : adds a relationship among the specified instances;
- **isRelated**(domain, property, range) : verifies if there is a specific kind of property among the given instances, returning a boolean value;
- **getInstances**(domain, property) : retrieves the instances that are targeted by the given domain and property, returning a Set<OWLNamedIndividual>;
- **addConcept**(concept) : adds the new concept in the ontology;
- **isSubConceptOf**(subConcept, superConcept) : verifies if the first concept is subclass of the second one, returning a boolean value; and
- **getConcepts**(instance) : retrieves a set of concepts for the given instance, returning a Set<OWLClass>.

IV. Usage Examples of the Ontology Artifact

We explain the use of our approach with a scenario commonly used in the agent literature: suppose a MAS which represents a soccer team and that each role is represented by concepts in an ontology. For example, a soccer team has players who can be right midfielders, which specialises the concept of midfield, which is a subclass of player, and so on. In certain moments the coach agent of a team needs to choose a player to replace other. To make its decision the coach agent just needs to look for the corresponding ontology concept and choose a player among the individuals of that concept.

A. Agent Decision Making using Ontology Information

Decision making is a process where an agent looks for the information available to it to decide which course of action to follow. This information generally comes from its environment perceptions, its initial beliefs, or from the communication with others agent, (i.e., beliefs from different sources). This work proposes an infrastructure layer in the form of a CArtAgO artifact to access domain specific knowledge provided by ontologies in a way that agents can use such information to make their decisions. Decision making is one example of how our approach may be employed, however it can be used in other domains where information and reasoning provided by ontologies is necessary or useful. In our example, the coach agent uses an ontology describing the team members and the roles of each agent/player in several situations (e.g., to retrieve information, reason and make its decisions).

This section shows examples of plans in Jason, which have the following format[1]: $triggering\_event$: $context$ <- $body$, where the $triggering\_event$ represents a new agent goal (or belief), and has a format such as $!goal(Parameter)$, the $context$ which defines the required preconditions to perform that plan, and the $body$ that is a sequence of actions and subgoals to fulfil that plan.

According to the soccer team scenario, the ontology concepts model soccer roles, such as Player, Midfield and Right Midfield (represented as concepts such as `C1`, `C2` and `C3`). The instances may represent Players, e.g., `i1` can be a player whose role is Right Midfield (concept `C3`). The instance `i1` can be related with `i2` through `r1` (e.g., `r1` can be defined as "is a player less defensive than"). Suppose an agent that needs to make a decision about which course of action to follow considering its context that is represented in an ontology. This decision can be guided, for example, by checking if a particular individual belongs to a particular concept. Using operations provided in the CArtAgO artifact presented in this paper to access the ontology, the agent can obtain the required information by executing the operation `isInstanceOf`, as shown in Figure 2. This operation returns a *boolean*, which is $true$ if the individual queried, `i1`, belongs to a given concept `C3`, or $false$ in the other case. The return unifies with the last parameter of the operation (`R`), which the agent uses to decide between executing `action_a1` or `action_a2`. Suppose a coach that needs to choose a player in some position, which is done by querying player agents that belongs to the desired role encoded as concepts in the ontology. For example, if in a given moment a player is injured, then the coach agent needs to scale another player in that position. To make this, the coach checks if a player agent belongs to the right midfield role. In this scenario, the coach has perceived that the injured player plays in front, but it does not remember its exact role (right midfield or left midfield). After checking this information, which is encoded as concepts in the ontology, the coach can make the decision of scaling a new player.

```
+!goal1: context
<-  isInstanceOf("i1","C3",R);
    if(R){ action_01; }
    else{ action_02; }.
```

Fig. 2. Jason plan using ontological knowledge (*isInstanceOf* operation).

Now, suppose that an agent needs to recover all individuals who participate in a particular relationship. The agent can use this information to make a decision about the existence of an individual in the returned set, or to select one of these individuals for a particular need. In this case, the operation *getInstances* can be used, as presented in Figure 3. The return of this operation is a set of individuals which have that relationship (`r1`) with the given instance (`i2`). Then, this plan tests if the set returned is empty, which leads to the execution of `action_03` if $true$, or in the other case the agent will pursue a goal involving a new decision making which uses the set of individuals returned (`goal3`). In our example, suppose that the coach wants to scale more defensive players to replace a particular player (`i2`) using the relation *defensive_substitution* (`r1`) which returns the list of defensive substitutions available for that player. If the list is empty, the coach may decide to reposition the players to have a more defensive team (`action_03`). In other case, where there is at least one player more defensive to substitute `i2`, the coach may choose one player of this set to be scaled (`!goal3(Set)`).

```
+!goal2: context
<-  getInstances("i2","r1",Set);
    if(.empty(Set)){ action_03; }
    else{ !goal3(Set); }.
```

Fig. 3. Jason plan using ontological knowledge (*getInstances* operation).

V. Comparing Ontology and Agent Approaches

An agent can represent its knowledge within its internal structures (e.g., its belief base), or in external structures (e.g., an ontology). This paper shows an artifact for agents to work with knowledge represented in ontologies, and such approach offers advantages in terms of expressiveness and reusability.

[1]We refer the reader to [4] for more details about the syntax and semantics of the language.

More expressiveness is obtained by the execution of semantic reasoners over the ontology; and more reusability comes from the possibility of different platforms updating and querying the same repository and formalism. Despite these improvements, programmers would be interested to known which approach is the fastest. So, we conducted an experiment to verify which approach presents better performance in terms of execution time.

To compare the ontology reasoning with the reasoning executed only in the agents, we defined ways to convert ontology statements to agent code, as depicted in Table I. These equivalences allow us to execute both approaches (which will return the same result) to compare their performance (i.e., the performance of reasoning with the ontology against simulating the same reasoning inside agents). Thus, the proposed artifact offers a new way to represent knowledge and new operations compared when using Jason alone and the proposed CArtAgO artifact to integrate agents with ontologies. For simplicity reason, Table I shows only the main statements which were used to test our approach of reasoning with the ontology in order to compare it with simulating the same reasoning only inside agents.

TABLE I
STATEMENTS IN ONTOLOGIES AND IN JASON CODE.

Statement	Ontology	Jason
x is instance of A	x : A	A(x)
x has property P targeting y	(x,y) : P	P(x,y)
B is subclass of A	B $\sqsubseteq$ A	A(x) :- B(x)
If B then A (B implies A)	B $\Rightarrow$ A	A :- B

A. Experiment Description

Our experiment compares the performance of executing agent plans that follow one of these two approaches for handling knowledge (internal or external structures). One approach uses our CArtAgO artifact to query information from ontologies; and the other approach queries the knowledge stored in the agent belief base. To access the corresponding performance impact, when using ontologies the number of individuals is increased, and when using the belief base the ontologies were converted to beliefs and rules in Jason. In both approaches we measured and compared the execution time for an agent to retrieve its information (from queries in an ontology or from its belief base).

The ontology used has 3 concepts (e.g., C1, C2 and C3) defined such as C3 is subclass of C2, and C2 is subclass of C1. The number of individuals ranges from 100 to 100.000, which are asserted to the most specific concept, in this case C3. The executed queries verify if an individual is an instance of the most specific (C3) and the most generic concept (C1). These queries were performed and compared both in ontology reasoning, and in Jason, and these queries return true, since an instance of C3 is inferred as C1 and the queried individuals were asserted as C3. All tests were executed in the same computer, which is a Mac Pro Server (OS X 10.9.4) with two 6-core Intel Xeon (2.4 GHz) CPU, 32 GB of RAM (DDR3 1333MHz) and 2 TB of disk storage. Regarding software, we used Java SDK 1.7 (build 1.7.0_65-b17), Jason 1.3.9, OWL API 2 version 3.50 and HermiT reasoner version 1.3.8.

The experiments measure the execution time of a Jason plan which uses an operation of our CArtAgO artifact (e.g., *isInstanceOf*). The time was measured for a hundred operations, and the sum of these values was divided by one hundred to obtain the average time of a single operation. We used this approach to obtain more accurate results by calculating an average that avoids spikes (too low or high values). This process was repeated ten times, and the final result is an average of these ten executions, each one executing a hundred of operations. The instances queried are selected based in the calculation of an interval ($interval = \frac{NumberOfInstances}{NumberOfQueries}$), where it is ensured that it will be selected members of all set, and not only members at the start or at the end of the set (uniformity).

B. Experiment Results

The results demonstrate that Jason performance can be improved by using this new approach instead of querying and reasoning with only the regular belief base. The execution time was measured to retrieve the same information, however in one case it is represented and retrieved from the ontology using our artifact, and in the other case it is stored and queried in the regular Jason belief base. When using ontologies, we tested two alternatives: with or without the execution of a semantic reasoner (respectively, Hermit and Structural). We queried for asserted, inferred and nonexistent knowledge. Our experiments demonstrate that the proposed approach enhance Jason performance, and also offers advantages of reuse and expressiveness. In applications with a low number of instances (Figure 4) we see a minor loss in performance, however if we consider instance rich ontologies and applications the proposed approach shows improvement (Figure 5). We would like to highlight that these two approaches are not mutually exclusive, which means that the agent programmer can choose to use

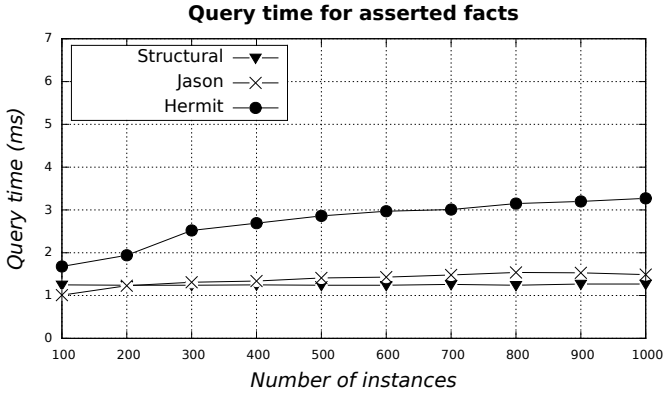

Fig. 4. Performance to retrieve asserted knowledge from a small number of individuals (instance of C3 axiom).

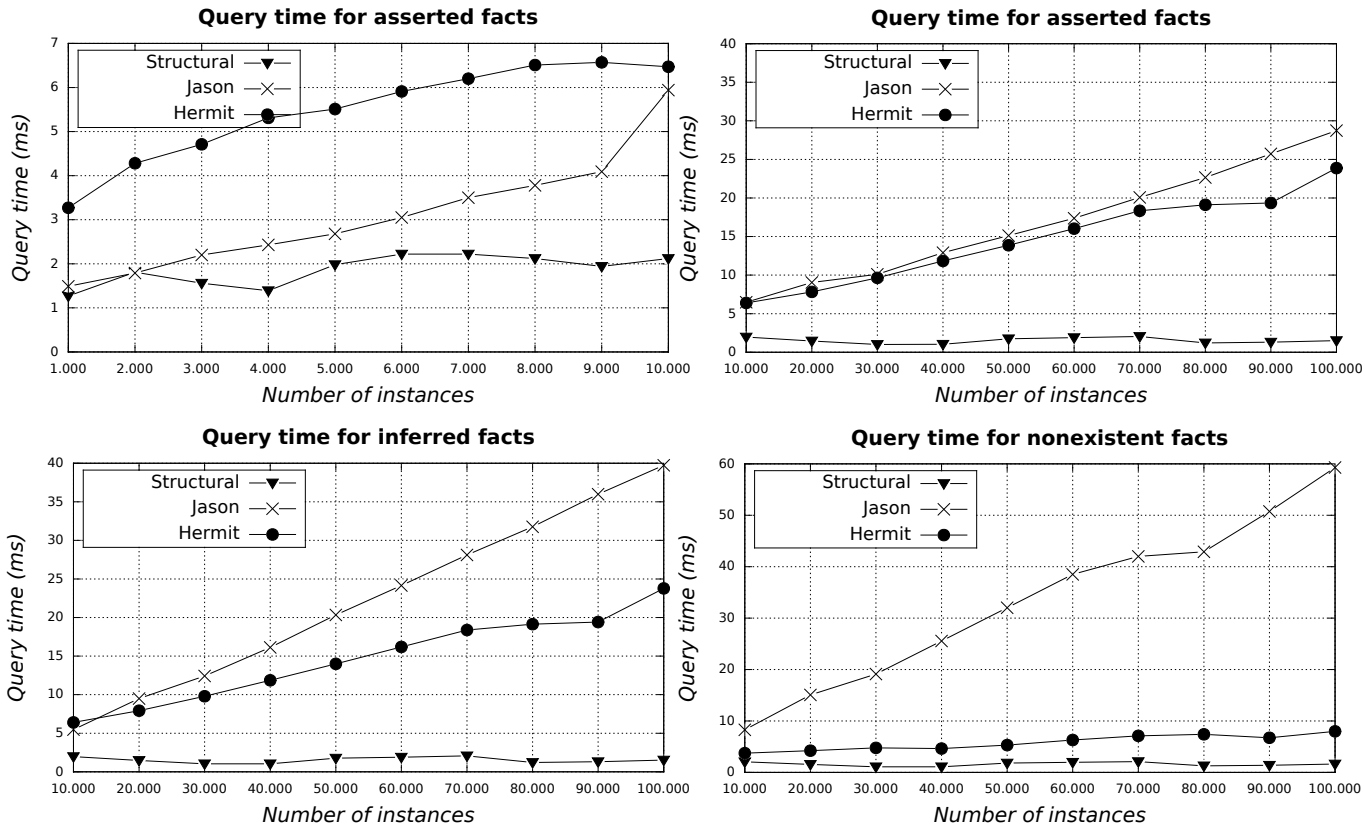

Fig. 5. Performance to retrieve asserted, inferred and nonexistent knowledge using ontology versus agent approaches.

them together, or just one if desired.

The experiments consider different sizes of ontologies, for example, Figure 4 shows our results with instances ranging from 100 to 1.000 instances. The results using ontologies with more instances (until 100.000 instances) are depicted in Figure 5. All tests demonstrate that the best performance is obtained when using our artifact with the Structural approach. When considering a large number of instances, the worst performance obtained comes from using Jason regular belief base. When retrieving inferred information (i.e., it is not explicit asserted), the performance of ontological approaches is similar for asserted facts. However, the regular belief base of Jason takes more time to apply the rules and return the result. When retrieving nonexistent information (which is not explicit asserted and cannot be inferred), the performance of ontological approaches is similar to previous ones. However, the regular belief base of Jason takes even more time than the previous cases.

VI. Related Work: Agents & Ontologies

AgentSpeak-DL [11] is an agent-oriented programming language that extends agents' belief base with Description Logic. The advantages of integrating ontologies with agents are: (*i*) more expressive queries in the belief base, since its results can be inferred from the ontology and thus are not limited to explicit knowledge; (*ii*) refined belief update given that ontological consistency of a belief addition can be checked; (*iii*) the search for a plan to deal with an event is more flexible (not limited to unification), i.e., subsumption relationships between concepts can be considered; and (*iv*) agents can share knowledge using ontology languages, such as OWL. AgentSpeak-DL extends agents' belief base with Description Logic in which the belief base includes: (*i*) one immutable TBox (terminological box, or conceptualisation) that characterises the domain concepts and properties; and (*ii*) one ABox (assertion box, or instantiation) with dynamic factual knowledge that changes according to the results of environment perception, plan execution and agent communication. AgentSpeak-DL approach enriches the agent belief base with the definition of complex concepts that can go beyond factual knowledge [11].

JASDL [12] implements AgentSpeak-DL in Jason to merge the agent belief base with ontological reasoning. It provides ontology manipulation capabilities to agents, (i.e., it is a practical approach for using ontologies and semantic reasoning in Jason agents). Agent programmers benefit from features such as plan trigger generalisation based on ontologies and the use of such knowledge in belief base querying. Jason modules were altered to implement JASDL, such as the belief base (that was extended to partly resides within an ontology ABox and a DL reasoner), the plan library and the agent architecture. JASDL provides reuse of ontological knowledge, new inferences that an agent can make based on

TABLE II
COMPARING RELATED WORK IN THE AREAS OF ONTOLOGIES AND MULTI-AGENT SYSTEMS.

Research	Overview of the work	Ontologies included	MAS platforms used
AgentSpeak-DL [11]	An approach for using ontologies during agent reasoning to extend agents' belief base with DL	It is a way for agents to represent knowledge and interact with ontologies	AgentSpeak
JASDL [12]	An implementation of AgentSpeak-DL in the Jason platform	Jason agents can represent knowledge and interact with ontologies	Jason
CooL-AgentSpeak [13]	An extension of AgentSpeak-DL with plan exchange and ontology services	Each agent has access only to its private ontologies	Jason
Our approach	A CArtAgO infrastructure to integrate multi-agent platforms with ontologies	Agents can access and manipulate shared ontologies using our artifact	Any platform supporting CArtAgO artifacts (e.g., Jason)

its beliefs, knowledge consistency, enhanced plan searching; and improved message processing with semantically-enriched inter-agent communication.

CooL-AgentSpeak [13] is an extension of AgentSpeak-DL with plan exchange and ontology services. It implements a CArtAgO artifact functioning as ontology repository which stores a possibly dynamic set of ontologies and offers ontology matching/alignment features. It searches for ontologically relevant plans not only in the agent's local plan library, but in the other agents' libraries too, according to a cooperation strategy (that is not based solely on unification and on the subsumption relation between concepts, but also on ontology matching). In short, CooL-AgentSpeak performs cross ontological unification for agents that do not disclose their ontologies to each other (that cooperate while preserving their privacy).

Our approach differs in some points. First, we implement an infrastructure layer which works as an interface between ontologies and MAS using a CArtAgO artifact that can be reused in several MAS platforms. On the other hand, AgentSpeak-DL [11] targets AgentSpeak, and JASDL [12] addresses Jason. CooL-AgentSpeak [13] also uses CArtAgO as a mean to integrate ontologies and agents, but our work differs from this one since we assume that agents may share their ontologies, while in CooL-AgentSpeak the agents do not share their ontologies. A comparison among such related work is depicted in Table II.

We have done previous work towards combining ontology and multi-agent technologies, whereby we developed tools to model multi-agent systems using an ontology as a meta-model [15]. That work extends our initial ideas towards models of multi-agent systems represented as abstractions in ontologies [16], [17]. We also developed an approach [18] for using an ontology to represent planning domains in HTN (Hierarchical Task Network). That approach can be used in developing plans for the Jason platform, given the similarity of HTN and Jason plans. However, our previous work does not address the use of ontologies as sources of information to be explored by agents.

VII. FINAL REMARKS

The integration of agent platforms with ontologies enables agents the ability to operate in a Semantic Web context. This work investigates how to enable current agent-oriented development platforms to transparently merge with such semantic technologies. As result, developers may obtain new features for developing complex software systems with a semantic infrastructure that applies software and knowledge engineering principles. The development of applications that integrate semantic and agent technologies is still an open challenge. To address this issue, we pointed out that ontology languages offering semantic querying and reasoning should be suitably integrated into agent development frameworks.

Our implementation to integrate ontologies within agents uses an artifact implemented in CArtAgO [2] that provides agents the ability to reason and manipulate ontologies. Our infrastructure is applicable to several agent-oriented platforms to engineer ontology-based AI applications, and we demonstrate how we use it in Jason [4] to access ontologies in OWL [5]. We measured the performance of our approach and compared with an alternative one which stores all the knowledge inside the agent, which demonstrated that the technology being proposed enable the development of new and more powerful AI applications. However, these approaches can be used together, in other words, an agent can represent part of its knowledge in its own belief base and part in ontologies to be accessed using the proposed artifact. As future work, we plan to carry out experiments to compare the use of our approach together with and against other agent platforms.

ACKNOWLEDGMENTS

Part of the results presented in this paper were obtained through research on a project titled "Semantic and Multi-Agent Technologies for Group Interaction", sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

REFERENCES

[1] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a practical OWL-DL reasoner," *Web Semant.*, vol. 5, no. 2, pp. 51–53, Jun. 2007.

[2] A. Ricci, M. Piunti, and M. Viroli, "Environment programming in multi-agent systems: An artifact-based perspective," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 2, pp. 158–192, Sep. 2011.

[3] A. R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Arguing About Task Reallocation Using Ontological Information in Multi-Agent Systems," in *12th International Workshop on Argumentation in Multiagent Systems*, 2015.

[4] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[5] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," W3C, Tech. Rep., February 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/

[6] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[7] M. Hadzic, P. Wongthongtham, T. Dillon, and E. Chang, *Ontology-based multi-agent systems*, ser. Studies in Computational Intelligence. Springer, 2009.

[8] F. Baader, I. Horrocks, and U. Sattler, "Description logics," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Springer, 2009, pp. 3–28.

[9] Q.-N. N. Tran and G. Low, "MOBMAS: a methodology for ontology-based multi-agent systems development," *Inf. Softw. Technol.*, vol. 50, no. 7-8, pp. 697–722, Jun. 2008.

[10] R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, *Multi-Agent Programming: Languages, Tools and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[11] A. F. Moreira, R. Vieira, R. H. Bordini, and J. F. Hübner, "Agent-oriented programming with underlying ontological reasoning," in *Proceedings of the 3rd international workshop on Declarative Agent Languages and Technologies*, ser. DALT'05. Springer-Verlag, 2006, pp. 155–170.

[12] T. Klapiscak and R. H. Bordini, "JASDL: a practical programming approach combining agent and semantic web technologies," in *The 6th international workshop on Declarative Agent Languages and Technologies*, vol. 5397. Springer, 2008, pp. 91–110.

[13] V. Mascardi, D. Ancona, M. Barbieri, R. H. Bordini, and A. Ricci, "CooL-AgentSpeak: Endowing AgentSpeak-DL agents with plan exchange and ontology services," *Web Intelligence and Agent Systems*, vol. 12, no. 1, pp. 83–107, 2014.

[14] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL ontologies," *Semant. web*, vol. 2, no. 1, pp. 11–21, Jan. 2011.

[15] A. Freitas, L. Hilgert, S. Marczak, F. Meneguzzi, R. H. Bordini, and R. Vieira, "A multi-agent systems engineering tool based on ontologies," in *34th International Conference on Conceptual Modeling, Stockholm, Sweden*, ser. Lecture Notes in Computer Science. Springer, 2015.

[16] A. Freitas, R. H. Bordini, F. Meneguzzi, and R. Vieira, "Towards integrating ontologies in multi-agent programming platforms," in *2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2013, Atlanta, Georgia, USA*, 2013.

[17] A. Freitas, D. Schmidt, A. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Applying ontologies and agent technologies to generate ambient intelligence applications," in *Joint Proceedings Collaborative Agents – Research & Development, CARE for Intelligent Mobile Services & Agents, Virtual Societies and Analytics*, 2014, pp. 22–33.

[18] ——, "Semantic representations of agent plans and planning problem domains," in *Engineering Multi-Agent Systems*, ser. Lecture Notes in Computer Science, F. Dalpiaz, J. Dix, and M. van Riemsdijk, Eds., vol. 8758. Springer International Publishing, 2014, pp. 351–366.