Explaining Semantic Reasoning Using Argumentation

Carlos Eduardo A. Ferreira[1], Alison R. Panisson[1(✉)], Débora C. Engelmann[2,4], Renata Vieira[3], Viviana Mascardi[4], and Rafael H. Bordini[2]

[1] Department of Computing, UFSC, Florianópolis, Brazil
alison.panisson@ufsc.br
[2] School of Technology, PUCRS, Porto Alegre, Brazil
debora.engelmann@edu.pucrs.br, rafael.bordini@pucrs.br
[3] CIDEHUS, University of Évora, Evora, Portugal
renatav@uevora.pt
[4] DIBRIS, University of Genoa, Genoa, Italy
viviana.mascardi@unige.it

Abstract. Multi-Agent Systems (MAS) are popular because they provide a paradigm that naturally meets the current demand to design and implement distributed intelligent systems. When developing a multi-agent application, it is common to use ontologies to provide the domain-specific knowledge and vocabulary necessary for agents to achieve the system goals. In this paper, we propose an approach in which agents can query semantic reasoners and use the received inferences to build explanations for such reasoning. Also, thanks to an internal representation of inference rules used to build explanations, in the form of argumentation schemes, agents are able to reason and make decisions based on the answers from the semantic reasoner. Furthermore, agents can communicate the built explanation to other agents and humans, using computational or natural language representations of arguments. Our approach paves the way towards multi-agent systems able to provide explanations from the reasoning carried out by semantic reasoners.

Keywords: Argumentation schemes · Multi-agent systems · Semantic reasoning · Explainability

1 Introduction

Explainability is pointed out as an essential characteristic in artificial intelligence applications, because it provides users with the necessary information for them to effectively understand, trust, and manage such applications [20]. The need for explaining a decision/reasoning/action was discussed as early as the 1970s, starting with the development of expert systems and the need for those systems to explain their decisions [1]. Nowadays, explainability becomes an essential

F. Dignum et al. (Eds.): PAAMS 2022, LNAI 13616, pp. 153–165, 2022.
https://doi.org/10.1007/978-3-031-18192-4_13

characteristic in MAS [40], given that MAS are one of the most powerful paradigms to implement complex distributed systems powered by artificial intelligence techniques.

Ontologies are known for empowering the execution of semantic reasoners, providing functionalities such as *consistency checking* [36]. Also, ontologies make it possible to share a common understanding of the structure of information among people and software agents as well as to reuse domain knowledge [17]. Integrating semantic reasoners and ontologies with agents enhances the knowledge representation features and reasoning capabilities of MAS applications [14]. Indeed, one notable possibility resulting from the use of ontologies in MAS is the capability of software agents to infer new knowledge based on logic rules [25] that can be applied by semantic reasoners. In this context, although semantic reasoners provide the computational steps (logic rules, concepts, etc.) used during inferences for an answer (i.e., query answering systems), they are not user-friendly explanations, and it would be difficult for users to understand those answers provided by semantic reasoners. Thus, in this paper, we propose an approach in which software agents are able to query semantic reasoners to obtain the answer, containing the trace of computational steps used for a particular inference, then they are able to translate those computational steps to explanations that can be communicated to software agents and humans, using computational and natural language representations of arguments. Our approach is based on the idea that inference rules, used by semantic reasoners to provide answers to queries, can be internally represented and stored by agents as argumentation schemes used to instantiate arguments based on the argumentation-based framework in [28,31].

The contributions of this work are: (i) we propose an approach that enables agents to query semantic reasoners and explain the received answer to other software agents or human users, as illustrated in Fig. 1; (ii) our approach allows the automatic translation of inference rules from answers received when querying semantic reasoners into argumentation schemes written in an Agent Oriented Programming Language (AOPL) (arrow 1 in Fig. 1). Translating those inferences into argumentation schemes, like those proposed by [39], bring us two main benefits. First, agents can store in their belief bases the argumentation schemes extracted from the answers obtained from the semantic reasoner and use them to instantiate arguments they can use for reasoning and communication (arrow 2 in Fig. 1). Second, using natural language templates for those argumentation schemes (arrows 3 and 4 in Fig. 1), agents can translate computational arguments into natural language arguments, using them to build natural language explanations for human users; and (iii) we describe a real-world multi-agent application for bed allocation in hospitals, exemplifying our approach based on what we have developed for that application.

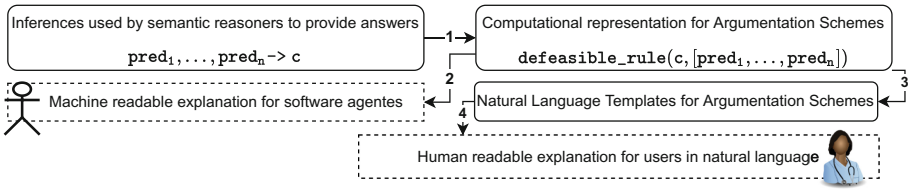

Fig. 1. Our approach for explaining semantic reasoning

2 Background

2.1 Agent Oriented Programming Languages

Among the many AOPLs and platforms, such as Jason, Jadex, Jack, Agent-Factory, 2APL, GOAL, Golog, and MetateM, as discussed in [3], we chose the Jason platform [4] to implement our work. Jason provides excellent conceptual/theoretical support; it extends the AgentSpeak language, an abstract logic-based AOPL introduced by Rao [32], which is one of the best-known languages inspired by the Beliefs, Desires, Intentions (BDI) architecture. Besides specifying BDI agents with well-defined mental attitudes, the Jason platform has some other features that are particularly interesting for our work, for example, strong negation, belief annotations, and (customisable) speech-act based communication.

2.2 Argumentation Schemes

Argumentation schemes represent reasoning/argument patterns normally found in daily conversation, as well as in specific argumentation, as scientific argumentation (scientific reports, discourses, etc.) [39]. Argumentation schemes provide a very elegant manner to represent and analyse these common argument patterns that are naturally found in the construction of reasoning.

For example, the *Argument from role to know in MAS* (*role to know* for short) from [28,31] is represented as follows:

> "Agent ag is currently playing a role R (its position) that implies knowing things in a certain subject domain S containing proposition A **(Major Premise)**. ag asserts that A (in domain S) is true (or false) **(Minor Premise)**. A is true (or false) **(Conclusion)**".

In order to allow agents to instantiate arguments from argumentation schemes, Panisson and colleagues [26–28,31] have proposed a framework to represent argumentation schemes in Jason multi-agent platform using defeasible inference rules. For example, the argumentation scheme *role to know* is represented in Jason as follows[1]:

[1] Note that argumentation schemes are modelled as agents beliefs, and the annotation `[as(as_name)]` is used to distinguish argumentation schemes from other beliefs.

```
defeasible_rule(Conclusion,[role(Agent,Role), role_to_know(Role,Domain),
  asserts(Agent,Conclusion),about(Conclusion,Domain)])[as(role_to_know)].
```

where the agents are able to instantiate such argumentation schemes with the information available to them and to evaluate the acceptability of the conclusion based on the interactions among such instantiated arguments [28,31].

For example, imagine that an agent `ag` knows that `john` (another agent in the system) is playing the role of `doctor`—`role(john, doctor)`. Further, `ag` knows that doctors know about cancer—`role_to_know(doctor, cancer)`. Therefore, if `john` asserts that "*smoking causes cancer*"—`asserts(john, causes(smoking, cancer))`, and given that causes of cancer are a subject matter related to cancer—`about(causes(smoking, cancer), cancer)`, `ag` is able to instantiate the argumentation scheme *role to know*, which allows `ag` to conclude that smoking causes cancer—`causes(smoking, cancer)`, based on the unification function $\{$`Agent` $\mapsto$ `john`, `Role` $\mapsto$ `doctor`, `Domain` $\mapsto$ `cancer`, `Conclusion` $\mapsto$ `causes(smoking, cancer)`$\}$.

Further, argumentation schemes combined with natural language templates can be used for translating arguments from a computational representation to natural language representation [29]. For example, the natural language template for the argumentation scheme `role_to_know` is as follows:

$\langle$"<Agent> is a <Role>, and <Role>s know about <Domain>. <Agent> asserts <Conclusion>, therefore we should believe that <Conclusion>".$\rangle_{[as(role\_to\_know)]}$

using the same unification function $\{$`Agent` $\mapsto$ `john`, `Role` $\mapsto$ `doctor`, `Domain` $\mapsto$ `cancer`, `Conclusion` $\mapsto$ `causes(smoking, cancer)`$\}$, it is possible to build the following natural language argument:

$\langle$"`john` is a `doctor`, and `doctor`s know about `cancer`. `john` asserts `smoking causes cancer`, therefore we should believe that `smoking causes cancer`".$\rangle_{[as(role\_to\_know)]}$

2.3 OWL Ontologies

An ontology is an explicit and formal specification of a shared conceptualisation consisting of concepts or classes, relationships, instances, attributes, axioms, restrictions, rules, and events [18,38]. Currently, ontologies are used in projects of several domains: Internet of Things (IoT) [22], smart cities [6], higher education [37], among others. A standard for representing ontologies that is widely used both in academia and industry is the OWL (Ontology Web Language), based on formal logic. OWL is based on description logic and has an inference mechanism based on this logic developed in the context of the global Semantic Web project and graphical editors for the creation of ontologies [38].

When developing an ontology, Semantic Web Rule Language (SWRL) [25] can be used to model more sophisticated inferences, and they are specified in the following format: $\texttt{pre}_1, \ldots, \texttt{pre}_n -> \texttt{conc}$, with $\texttt{pre}_1, \ldots, \texttt{pre}_n$ the n premises of the rule, and `conc` the conclusion of the rule.

3 Scenario

In this paper, we use a scenario of bed allocation in hospitals, based on the work reported in [8,10–12], in order to exemplify how we have built our approach for explaining semantic reasoning using argumentation. Bed allocation is a challenge hospitals face since hospital beds are scarce, and when poorly managed, it can generate long lines and chaos in emergency rooms [19]. The large number of constraints that need to be considered during the allocation process makes this process difficult for humans to perform. Considering that MAS can be powered by different artificial intelligence techniques, they are suitable to deal with complex problems like bed allocation in hospitals. Thanks to recent developments, MAS are equipped with communication in natural language [9,10] and ontological reasoning [14].

Below, we have an example of dialogue that occurs in the multi-agent application developed, in which a human operator tries to allocate a patient named `patient2` to the bed `101b`; however, bed `101b` is considered unsuitable for that patient given the hospital rules modelled in the domain-specific ontology of that application. Thus, the agent (software agent) responsible to assist the bed allocation in the multi-agent application opens a dialogue with the human operator to inform that bed is not suitable for that patient, later explaining why it is not suitable:

- **Assistant**: Your bed allocation plan has flaws. Bed `101b` is unsuitable for `patient2`.
- **Operator**: Why do you think bed `101b` is unsuitable for **patient2**?
- **Assistant**: Because patient `patient1` is of care `semi-Intensive-Care` and occupies bed `101a`. So bed `101a` is of care `semi-Intensive-Care`. Bed **101a** is in bedroom `101` and it is of care `semi-Intensive-Care`. So bedroom `101` is of care `semi-Intensive-Care`. Bed **101b** is in bedroom `101` and bedroom `101` is of care `semi-Intensive-Care`. So bed `101b` is of care `semi-Intensive-Care`. Patient `patient2` is of care `minimal-Care` and bed `101b` is of care `semi-Intensive-Care` that is different from `minimal-Care`. So bed `101b` is unsuitable for patient `patient2`.
- **Operator**: And how about bed 103a? Is it suitable?
- **Assistant**: No, it isn't.
- **Operator**: Why?
- **Assistant**: Because patient `patient3` is of care `intensive-Care` and occupies bed `103b`. So bed `103b` is of care `intensive-Care`. Bed `103b` is in bedroom `103` and is of care `intensive-Care`. So bedroom `103` is of care `intensive-Care`. Bed `103a` is in bedroom `103` and bedroom `103` is of care `intensive-Care`. So bed `103a` is of care `intensive-Care`. Patient `patient2` is of care `minimal-Care` and bed `103a` is of care `intensive-Care` that is different from `minimal-Care`. So bed `103a` is unsuitable for patient `patient2`.

This dialogue in natural language is resulting from the approach we will present in this paper, combined with other technologies: (i) Dial4JaCa[2] [10],

[2] https://github.com/smart-pucrs/Dial4JaCa.

which allows the integration between chatbot technologies and MAS, and it is used to identify users' intentions and to extract entities from natural language inputs as well as to provide responses to users in natural language; and (ii) an interface between MAS and ontologies [14], which allows extending the agents' belief bases with semantic technologies. While Dial4JaCa represents an important component in the multi-agent application developed, which provides an interactive interface with human users, in this work we are going to focus only on the the components necessary to present our approach, namely the interface between ontologies and MAS.

4 Querying Ontologies

In a series of papers, Freitas and colleagues [13–15] developed an approach to interface OWL ontologies and MAS using CArtAgO artifacts [33]. Using the proposed approach, agents are able to store, access, and query domain-specific OWL ontologies. The infrastructure proposed by [14] has been used in different applications as evidenced by [30,35], and it provides an elegant architecture for engineering intelligent systems based on the MAS paradigm.

While agents are able to query OWL ontologies using the approach developed in [14], the interface does not provide explanations for queries, or even traces of computational steps used by the semantic reasoner to reach that particular conclusion. Even though an agent would have access to answers from semantic reasoners, queries result from complex reasoning executed over domain-specific inference rules, modelled using SWRL rules [25], and those inferences would not be easily understood by software agents in their original form[3].

We have extended the approach presented by [14] to process the traces of computational steps (including the application of inference rules) used by semantics reasoners during queries to OWL ontologies. Then, using our approach to translate those answers into an agent-oriented programming representation, agents are able not only to understand and manipulate that information but also to build explanations from external semantic reasoning.

5 Translating SWRL Rules into Argumentation Schemes

Our approach for building explanations for answers from semantic reasoners is based on the idea that agents are able to internally model and store inference rules returned by semantic reasoners (when providing an answer for a query) using argumentation schemes like those presented in Sect. 2.2, which are processed by agents using the framework presented in [28,31]. Then, using templates in natural language for argumentation schemes, also introduced in Sect. 2.2, agents build natural language explanations for those answers.

First, we provide an approach for agents to query semantic reasoners, obtaining the answer for the queries in the format of traces of computational steps

[3] Frequently, they are not easily understood even by users of those technologies.

Table 1. Correspondence between answers from semantic reasoners and an AOPL representation

Answer from the Semantic Reasoner	Representation in AOPL
101b is-in 101	`is_in("101b","101")`
DifferentIndividuals: Intensive-Care, Minimal-Care, Semi-Intensive-Care	`isDifferentFrom("Intensive-Care","Minimal-Care")` `isDifferentFrom("Intensive-Care","Semi-Intensive-Care")` `isDifferentFrom("Minimal-Care","Semi-Intensive-Care")`
101a is-in 101	`is_in("101a","101")`
Patient2 is-care Minimal-Care	`is_care("Patient2","Minimal-Care")`
101b **Type** Hospital_Bed	`hospital_Bed("101b")`
Hospital_Bed(?B1r), Bedroom(?Br), is-in(?B1r,?Br), bed-is-care(?B1r,?C1r) -> bedroom-is-care (?Br,?C1r)	`defeasible_rule(bedroom_is_care(Br,C1r), [hospital_Bed(B1r),bedroom(Br),is_in(B1r,Br), bed_is_care(B1r,C1r)])[as(<schemeName>)]`
101a **Type** Hospital_Bed	`hospital_Bed("101a")`
Patient(?P2r), Hospital_Bed(?B2r), is-care(?P2r,?C2r), bed-is-care(?B2r,?C1r), **DifferentFrom**(?C1r,?C2r) -> is-unsuitable-for(?B2r,?P2r)	`defeasible_rule(is_unsuitable_for(B2r,P2r)[patient(P2r), hospital_Bed(B2r),is_care(P2r,C2r),bed_is_care(B2r,C1r), differentFrom(C1r,C2r)])[as(<schemeName>)]`
101 **Type** Bedroom	`bedroom("101")`
Hospital_Bed(?B2r), Bedroom(?Br), is-in(?B2r,?Br), bedroom-is-care(?Br,?C1r) -> bed-is-care(?B2r,?C1r)	`defeasible_rule(bed_is_care(B2r,C1r), [hospital_Bed(B2r),bedroom(Br),is_in(B2r,Br), bedroom_is_care(Br,C1r)])[as(<schemeName>)]`
Patient1 occupy-one 101a	`occupy_one("Patient1","101a")`
Patient1 is-care Semi-Intensive-Care	`is_care("Patient1","Semi-Intensive-Care")`
Patient2 **Type** Patient	`patient("Patient2")`
Patient(?P1r), is-care(?P1r,?C1r), Hospital_Bed(?B1r), occupy-one(?P1r,?BIr) -> bed-is-care(?BIr,?C1r)	`defeasible_rule(bed_is_care(B1r,C1r), [patient(P1r),is_care(P1r,C1r),hospital_Bed(B1r), occupy_one(P1r,B1r)])[as(<schemeName>)]`
Patient1 **Type** Patient	`patient("Patient1")`

(including concepts, classes and inference rules) used by the semantic reasoner to infer that particular query. We implemented our approach using a CArtAgO artifact [34], using the OWL API[4] (a Java API for creating, manipulating and serializing OWL Ontologies) as a basis for querying ontologies in conjunction with Openllet[5] (an open-source OWL DL reasoner for Java) to extract the answers about inferences made based on SWRL rules.

Through the CArtAgO artifact developed, we provide agents with an operation called `getExplanation` that receives as a parameter the string corresponding to the `objectProperty` (e.g. `"is-unsuitable-for"`) that relates the individuals, and the predicate corresponding to the query (e.g. `is_unsuitable_for ("101b","patient2")`). Then, the artifact executes the query to the semantic reasoner, and provides the answer to agents in the following format:

```
explanationTerms(rules(RulesList),assertions(AList),classInfo(CInfoList))
```

To build this internal representation based on the data returned for OWL API and Openllet we created a class that converts each axiom to an AOPL

[4] https://github.com/owlcs/owlapi.
[5] https://github.com/Galigator/openllet.

representation. An example of this process is shown in Table 1, based on the answer received by the assistant agent from the running scenario presented in Sect. 3. Also, our approach translates the inference rules returned into an answer to the format of argumentation schemes. This representation allows agents to build arguments from the reasoning patterns extracted from the answers, being able to reason, understand and communicate arguments instantiated from these argumentation schemes using the argumentation-based framework presented in [31].

We also created an internal action named `unifyRule` that receives as parameters the rule list and the assertion list that we identified with the logical variables `RulesList` and `AList`, respectively, in the `explanationTerms` internal representation introduced above. It allows agents to unify terms in argumentation schemes based on the assertions received in the answer provided by our interface. That is, the unification function is obtained from `RulesList`, `AList`, and `CInfoList`. This process provides agents with the set of arguments extracted from the answer, which we call here an *argumentation-based explanation*. With an internal representation of those reasoning patterns, agents are able to build and communicate explanations represented in a computational representation for arguments, which is useful when the system requires agents to provide explanations to other software agents.

In order to provide explanations to human users, agents use natural language templates for argumentation schemes [29] to translate those arguments to natural language arguments, then using those arguments to build and provide natural language explanations to human users.

Continuing our example, below, we demonstrate 1 of the 40 domain-specific rules[6] modelled by experts in order to establish the reasoning pattern necessary to bed allocation in the multi-agent application described in Sect. 3. These rules are used by the semantic reasoner to provide the answer used by the assistant agents when building the explanation from our running scenario.

> **Argumentation Scheme for Unsuitable Beds (AS4UB)**: "Patient `P` is of care `C1` (**premise**). Bed `B` is of care care `C2` (**premise**). Care `C1` is different of care `C2` (**premise**). Bed `B` is unsuitable for patient `P` (**conclusion**)".

This argumentation scheme is extracted from the below SWRL rule available in the ontology used in the multi-agent application.

```
Patient(?P), Hospital_Bed(?B), is-care(?P,?C1), bed-is-care(?B,?C2),
DifferentFrom(?C1,?C2) ->  is-unsuitable-for(?B,?P)
```

When the assistant agent queries the semantic reasoners, asking if a particular bed `101b` is unsuitable for the patient `patient2` – `is_unsuitable_for(101b, patient2)` – looking for validating the operator allocation, the semantic reasoner

[6] All rules are available at https://github.com/DeboraEngelmann/explaining-ontological-reasoning/blob/main/base_rules.md.

will answer that query with the trace of computational steps used to make the inference. From the answer provided by the semantic reasoner, our approach automatically translates the inference rules contained in that answer to argumentation schemes, according the representation required by the argumentation-based framework from [31], i.e., using defeasible inference rules represented by the predicate `defeasible_rule(Conclusion,Premises)`, in which `Conclusion` represents the conclusion of the rule, and `Premises` the set of premises used in the body of that particular rule. For example, the argumentation scheme presented in this section is internally represented by agents as follows:

```
defeasible_rule(is_unsuitable_for(B,P),  [patient(P), hospital_Bed(B),
is_care(P,C1), bed_is_care(B,C2), differentFrom(C1,C2)])[as(as4ub)]
```

Thus, when agents need to communicate an explanation to another software agent, for example, to explain why bed `101b` is unsuitable to patient `patient2`, according to our running scenario, they are going to build an explanation using the computational representation for arguments introduced in Sect. 2.2, using the argumentation-based framework [28,31].

```
explanation(is_unsuitable_for(101b,patient2),
[defeasible_rule(bed_is_care(101a,semi-intensive-care),[...])[as(as4bc1)],
defeasible_rule(bedroom_is_care(101,semi-intensive-care),[...])[as(as4br)],
defeasible_rule(bed_is_care(101b,semi-intensive-care),[...])[as(as4bc2)],
defeasible_rule(is_unsuitable_for(101b,patient2), [patient(patient2),
  hospital_Bed(101b), is_care(patient2,minimal-care),
  bed_is_care(101b,semi-intensive-care),
  differentFrom(minimal-care,semi-intensive-care)])[as(as4ub)]])}
```

To build the explanation presented above[7], agents query their belief base for the predicate they are interested to provide an explanation for, using `argument(Q, Arg)` with `Q` the queried predicate, and `Arg` a free variable that will unify with the argument supporting `Q`. This query uses the argumentation-based framework [31], which looks for argumentation schemes that infer that particular queried information, using the information available to the agent to instantiate argumentation schemes, building an argument that supports the queried information. In our scenario, `Arg` unifies with the set of arguments (or chained/complex argument) presented above, supporting `is_unsuitable_for(101b,patient2)`.

When it is necessary to communicate with human users, agents are able to build natural language explanations, translating the computational representation of arguments to natural language arguments, using natural language templates for argumentation schemes, as we describe in the next section.

5.1 Translating Arguments to Natural Language Explanations

When agents need to communicate an explanation in natural language, they use the plan `+!translateToNaturalLanguage` that implements how agents translate argu-

[7] We omitted the premises of argumentation schemes we did not present in this paper. All argumentation schemes are available in the GitHub repository.

ments from a computational representation to natural language, then aggregating those natural language arguments into an explanation. As it can be observed in Sect. 3, an explanation might be a sequence of arguments (also considered as a chained/complex argument). Thus `+!translateToNaturalLanguage` receives a list of one or more arguments (each one of those arguments are instances of an argumentation scheme), and then it translates each computational argument to a corresponding natural language argument, recovering the natural language template to the argumentation scheme used to instantiate that particular argument and returning its natural language representation.

6 Related Work and Conclusions

Explainablility has become a central topic in AI, and the literature of MAS exploring explainability has significantly increased in the last few year, mostly of it focusing on the health domain [2,5,7,16,21]. Also, interesting work exploring the integration of ontologies and MAS has been developed, for example, AgentSpeakDL [24], and CooL-AgentSpeak [23].

While our approach is inspired by much of these works, to the best of our knowledge, our approach is the first to propose that intelligent agents are able to explain the reasoning executed externally by semantic reasoners. Also, our approach is the first to propose an internal representation for SWRL rules using argumentation schemes in MAS, which allows agents to understand and manipulate those reasoning patterns that were only processed by semantic reasoners at the ontology level.

In this paper, we proposed an approach in which agents are able to build explanations based on answers received from semantic reasoners about their queries. Explanations can be communicated to both software agents and human users, using a computational or a natural language representation for explanations, respectively.

Our approach is based on the idea that inference rules contained in answers obtained from queries to semantic reasoners can be translated into argumentation schemes that agents are able to understand and manipulate using the argumentation-based framework from [28,31]. Thus, agents can instantiate arguments from argumentation schemes, using them for reasoning and communication. Furthermore, using natural language templates for argumentation schemes, agents are able to translate the computational representation of arguments into natural language arguments. Thus, they are able not only to build explanations for other software agents but also for human users.

Combining the approach presented in this work with other technologies used for human-agent interactions [10] allowed us to achieve the natural language dialogue presented in Sect. 3, which is part of a real-world multi-agent application (under development) that aims to help human operators in the process of bed allocation in hospitals.

Acknowledgements. This research was partially funded by CNPq and CAPES.

References

1. Akata, Z., et al.: A research agenda for hybrid intelligence: augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence. Computer **53**(8), 18–28 (2020)
2. Baskar, J., Janols, R., Guerrero, E., Nieves, J.C., Lindgren, H.: A multipurpose goal model for personalised digital coaching. In: Montagna, S., Abreu, P.H., Giroux, S., Schumacher, M.I. (eds.) A2HC/AHEALTH 2017. LNCS (LNAI), vol. 10685, pp. 94–116. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70887-4_6
3. Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F.: Multi-Agent Programming: Languages, Tools and Applications. Springer, Heidelberg (2009)
4. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, Hoboken (2007)
5. Cheng, C.Y., Qian, X., Tseng, S.H., Fu, L.C.: Recommendation dialogue system through pragmatic argumentation. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication, pp. 335–340. IEEE (2017)
6. De Nicola, A., Villani, M.L.: Smart city ontologies and their applications: a systematic literature review. Sustainability **13**(10), 5578 (2021)
7. Donadello, I., Dragoni, M., Eccher, C.: Explaining reasoning algorithms with persuasiveness: a case study for a behavioural change system. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 646–653 (2020)
8. Engelmann, D., Couto, J., Gabriel, V., Vieira, R., Bordini, R.: Towards an ontology to support decision-making in hospital bed allocation. In: Proceedings of 31st International Conference on Software Engineering & Knowledge Engineering, pp. 71–74 (2019)
9. Engelmann, D., et al.: Dial4JaCa – a demonstration. In: Dignum, F., Corchado, J.M., De La Prieta, F. (eds.) PAAMS 2021. LNCS (LNAI), vol. 12946, pp. 346–350. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85739-4_29
10. Engelmann, D., et al.: Dial4JaCa – a communication interface between multi-agent systems and chatbots. In: Dignum, F., Corchado, J.M., De La Prieta, F. (eds.) PAAMS 2021. LNCS (LNAI), vol. 12946, pp. 77–88. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85739-4_7
11. Engelmann, D.C., Cezar, L.D., Panisson, A.R., Bordini, R.H.: A conversational agent to support hospital bed allocation. In: Britto, A., Valdivia Delgado, K. (eds.) BRACIS 2021. LNCS (LNAI), vol. 13073, pp. 3–17. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91702-9_1
12. Engelmann, D.C.: An interactive agent to support hospital bed allocation based on plan validation. Dissertation, PUCRS (2019)
13. Freitas, A., Panisson, A.R., Hilgert, L., Meneguzzi, F., Vieira, R., Bordini, R.H.: Integrating ontologies with multi-agent systems through CArtAgO artifacts. In: 2015 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) (2015)
14. Freitas, A., Panisson, A.R., Hilgert, L., Meneguzzi, F., Vieira, R., Bordini, R.H.: Applying ontologies to the development and execution of multi-agent systems. In: Web Intelligence, vol. 15, pp. 291–302. IOS Press (2017)
15. Freitas, A., Schmidt, D., Panisson, A., Bordini, R.H., Meneguzzi, F., Vieira, R.: Applying ontologies and agent technologies to generate ambient intelligence applications. In: Koch, F., Meneguzzi, F., Lakkaraju, K. (eds.) AVSA CARE 2014. CCIS, vol. 498, pp. 22–33. Springer, Cham (2014). https://doi.org/10.1007/978-3-662-46241-6_3

16. Grando, A., Moss, L., Bel-Enguix, G., Jiménez-López, M.D., Kinsella, J.: Argumentation-based dialogue systems for medical training. In: Neustein, A., Markowitz, J. (eds.) Where Humans Meet Machines, pp. 213–232. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-6934-6_10
17. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
18. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum. Comput. Stud. **43**(5), 907–928 (1995)
19. da Silveira Grübler, M., da Costa, C.A., Righi, R., Rigo, S., Chiwiacowsky, L.: A hospital bed allocation hybrid model based on situation awareness. Comput. Inform. Nurs. **36**, 249–255 (2018)
20. Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., Yang, G.Z.: XAI-explainable artificial intelligence. Sci. Robot. **4**(37) (2019)
21. Kökciyan, N., et al.: A collaborative decision support tool for managing chronic conditions. In: MedInfo, pp. 644–648 (2019)
22. Li, W., Tropea, G., Abid, A., Detti, A., Le Gall, F.: Review of standard ontologies for the web of things. In: 2019 Global IoT Summit (GIoTS), pp. 1–6 (2019)
23. Mascardi, V., Ancona, D., Bordini, R.H., Ricci, A.: CooL-AgentSpeak: enhancing AgentSpeak-DL agents with plan exchange and ontology services. In: 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 109–116. IEEE (2011)
24. Moreira, Á.F., Vieira, R., Bordini, R.H., Hübner, J.F.: Agent-oriented programming with underlying ontological reasoning. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS (LNAI), vol. 3904, pp. 155–170. Springer, Heidelberg (2006). https://doi.org/10.1007/11691792_10
25. O'Connor, M.: The semantic web rule (2009)
26. Panisson, A.R., Bordini, R.H.: Knowledge representation for argumentation in agent-oriented programming languages. In: 2016 Brazilian Conference on Intelligent Systems, BRACIS (2016)
27. Panisson, A.R., Bordini, R.H.: Uttering only what is needed: enthymemes in multi-agent systems. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 1670–1672. International Foundation for Autonomous Agents and Multiagent Systems (2017)
28. Panisson, A.R., Bordini, R.H.: Towards a computational model of argumentation schemes in agent-oriented programming languages. In: International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) (2020)
29. Panisson, A.R., Engelmann, D.C., Bordini, R.H.: Engineering explainable agents: an argumentation-based approach. In: Alechina, N., Baldoni, M., Logan, B. (eds.) EMAS 2021. LNCS, vol. 13190, pp. 273–291. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-97457-2_16
30. Panisson, A.R., et al.: Arguing about task reallocation using ontological information in multi-agent systems. In: 12th International Workshop on Argumentation in Multiagent Systems, vol. 108 (2015)
31. Panisson, A.R., McBurney, P., Bordini, R.H.: A computational model of argumentation schemes for multi-agent systems. Argument Comput. 1–39 (2021)
32. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J.W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0031845
33. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Auton. Agent. Multi-Agent Syst. **23**(2), 158–192 (2011)

34. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: an infrastructure for engineering computational environments in MAS. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) 3rd International Workshop "Environments for Multi-Agent Systems" (E4MAS), pp. 102–119 (2006)
35. Schmidt, D., Panisson, A.R., Freitas, A., Bordini, R.H., Meneguzzi, F., Vieira, R.: An ontology-based mobile application for task managing in collaborative groups. In: Florida Artificial Intelligence Research Society Conference (2016)
36. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. J. Web Semant. **5**(2), 51–53 (2007)
37. Tapia-Leon, M., Rivera, A.C., Chicaiza, J., Luján-Mora, S.: Application of ontologies in higher education: a systematic mapping study. In: 2018 IEEE Global Engineering Education Conference (EDUCON), pp. 1344–1353 (2018)
38. Vieira, R., Abdalla, D.S., Silva, D.M., Santana, M.R.: Web Semântica: Ontologias, Lógica de Descrição e Inferência, pp. 127–167. SBC (2005)
39. Walton, D., Reed, C., Macagno, F.: Argumentation Schemes. Cambridge University Press, Cambridge (2008)
40. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley, Hoboken (2009)