Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Intentional dialogues in multi-agent systems based on ontologies and argumentation

by

Débora Cristina Engelmann

Theses Series **DIBRIS-TH-2022-XX**

DIBRIS, Università di Genova

Via Opera Pia, 13 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,

Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum

Intentional dialogues in multi-agent systems based on ontologies and argumentation

by

Débora Cristina Engelmann

December, 2022

Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi
Indirizzo Informatica
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Univ. di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
http://www.dibris.unige.it/

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum
(S.S.D. INF/01)

Submitted by Débora Cristina Engelmann
DIBRIS, Univ. di Genova
. . . .

Date of submission:
December 2022

Title:
Intentional dialogues in multi-agent systems based on ontologies and argumentation

Advisors:
Prof. Dr. Viviana Mascardi and
Prof. Dr. Rafael Heitor Bordini

Co-Advisors:
Prof. Dr. Angelo Ferrando and
Prof. Dr. Alison Roberto Panisson

Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università di Genova
. . .

Ext. Reviewers:
Prof. Dr. Roberta Calegari and
Prof. Dr. Serena Villata

Abstract

Some areas of application, for example, healthcare, are known to resist the replacement of human operators by fully autonomous systems. It is typically not transparent to users how artificial intelligence systems make decisions or obtain information, making it difficult for users to trust them. To address this issue, we investigate how argumentation theory and ontology techniques can be used together with reasoning about intentions to build complex natural language dialogues to support human decision-making. Based on such an investigation, we propose MAIDS, a framework for developing multi-agent intentional dialogue systems, which can be used in different domains. Our framework is modular so that it can be used in its entirety or just the modules that fulfil the requirements of each system to be developed. Our work also includes the formalisation of a novel dialogue-subdialogue structure with which we can address ontological or theory-of-mind issues and later return to the main subject. As a case study, we have developed a multi-agent system using the MAIDS framework to support healthcare professionals in making decisions on hospital bed allocations. Furthermore, we evaluated this multi-agent system with domain experts using real data from a hospital. The specialists who evaluated our system strongly agree or agree that the dialogues in which they participated fulfil Cohen's desiderata for task-oriented dialogue systems. Our agents have the ability to explain to the user how they arrived at certain conclusions. Moreover, they have semantic representations as well as representations of the mental state of the dialogue participants, allowing the formulation of coherent justifications expressed in natural language, therefore, easy for human participants to understand. This indicates the potential of the framework introduced in this thesis for the practical development of explainable intelligent systems as well as systems supporting hybrid intelligence.

Table of Contents

Chapter 1

Introduction

The research reported in this thesis focuses on combining Argumentation Theory and Ontology techniques to support complex dialogues in natural language. In particular, we have created an approach to support the development of dialogue systems that take advantage of that combination of techniques within BDI (Belief-Desire-Intention) agents to assist humans in decision making. We have made this approach adaptable enough to be applied across multiple domains.

Many areas can benefit from a decision support system. In the healthcare area, for example, bed allocation represents a challenge to hospitals (especially in developing countries such as Brazil) because hospital beds are a scarce resource. Also, hospital environments are highly dynamic and uncertain, so allocating hospital beds optimally plays an essential role in the overall planning of hospital resources [TEDF$^+$12]. A system that suggests better bed allocations for the professional responsible for this task is interesting in this context.

Moreover, the healthcare area is known to have a certain resistance to the replacement of human operators by automated systems. This is understandable because it deals with humans, so a wrong decision by a fully autonomous system – that is, a system that also takes the final decision on its own – raises legal issues that most countries cannot yet cope with (who is accountable for a wrong decision? The engineer that implemented the tool? The data scientist that trained it? ...). That is why it is important to have a system that assists decision making but where the human operator makes the final decision. Also, we need to consider that if the system cannot explain the suggested decisions, it is possible that the user will not understand and, because of that, ignore the suggestion. Another important challenge is maintaining and increasing the user's willingness to interact with the technical system [NM17]. In these cases, a mixed-initiative system, which supports human-computer interaction, becomes useful [HLF04]. In the context of Hybrid Intelligence (HI) [ABdR+20], it requires humans and intelligent systems to work together, and one of the key challenges to achieving this partnership is the capability of agents to understand human actors.

The Multi-Agent Intentional Dialogue System (MAIDS) framework we present in this thesis focuses on combining Argumentation Theory techniques [PB17a, FRS18, dOGPB+20], Ontology [Gru95], and Theory of Mind (ToM) [G+12] to support complex dialogues in natural language. MAIDS includes components to support the development of complex Multi-Agent System (MAS) applications, such as: (i) support for the creation of dialogues in natural language to facilitate the interaction with human operators; (ii) argumentation-based reasoning and dialogues which allow agents to reason about and communicate well-supported information; (iii) ontologies to help agents organise domain knowledge and perform semantic reasoning; and (iv) theory of mind to allow agents to infer other agents mental attitudes leading to purposeful communication. All these components are developed on top of an Agent-Oriented Programming Language (AOPL) based on the BDI architecture, which provides a suitable basis for all those techniques.

1.1 Motivation

In some areas, there is resistance to replacing human operators with fully autonomous systems. It is typically not transparent to users how artificial intelligence systems make decisions or obtain information [HTFJ19], making it difficult for users to trust them. To address this issue, we have focused our efforts on developing conversational agents to assist humans in decision-making. Our approach uses Argumentation Theory and Ontology techniques as the basis for reasoning in natural language dialogues. Using these techniques, the decisions can be made more transparent and explainable to users, with more natural and richer dialogue structures. Furthermore, our approach is adaptable enough to be applied across multiple domains, but we have focused our evaluations on Healthcare.

In the healthcare area, effective management of hospital beds has been the focus of much research, such as the IMBEDS model that uses artificial neural networks and multiattribute value theory for decision-making [GdCR+18a]; statistical and data mining approach [TEDF+12]; optimisation model with an evolutionary algorithm for bed allocation [eOdVAP20]; and also reviews of the literature were carried out [MR11, AJSS17]. Although all the work mentioned above seeks to improve bed management, they do not provide natural language interaction and do not offer decision support for the professional having complete control over the allocations, nor are the decisions explainable. Also, there are still few studies that practically apply the formal argumentation-based models of reasoning and dialogue to support interactions between software and human users [ERS+18]. In addition, there is a lack of work that empirically evaluates such models with human users [ERS+18].

1.2 Objectives

Motivated by the context discussed above, we identified the following general objective: *to investigate how argumentation theory and ontology techniques can be used together with reasoning about intentions to build complex natural language dialogues to support human decision making.*

Our specific objectives are:

- Design and formalise an approach to argumentation-based dialogues and ontological reasoning.
- Construct an architecture/framework for developing explainable systems and implement all the necessary components of the architecture.
- Apply the proposed formal model to a real-world domain and problem, such as healthcare, and fully implement a dialogue system based on that formalisation using the framework.
- Evaluate the approach with domain experts based on Cohen's desiderata for task-oriented dialogue systems [Coh19].

1.3 Main Contributions

Among the contributions of this thesis we highlight: (i) identification of the ways that the scientific community have used argumentation techniques to achieve explainable artificial intelligence in dialogue systems included in a literature survey published in [EDP+22]; (ii) development of the MAIDS framework [EPV+] to support the development of explainable dialogue systems based on BDI agents to assist humans in decision making; (iii) introduction and formalisation of a multi-part belief base for a BDI agent programming language and a structured approach to dialogues where agents argue about the information from the main belief base component but can move on to subdialogues to discuss specific issues related to the ontological component or the ToM component of the multi-part belief base; (iv) creation of the Dial4JaCa framework [EDK^{+}21b, EDK^{+}21a] to enable intelligent agents to communicate with humans through natural-language interaction; (v) creation of the Onto4JaCa framework to give intelligent agents the ability to use and manage the information contained in ontologies during their reasoning processes [FPE^{+}22]; (vi) creation of the RV4JaCa framework [EFP^{+}22] which supports the use of runtime verification in multi-agent systems developed on the JaCaMo platform [BBHR20]; (vii) implementation of an explainable system based on the MAIDS framework to assist in decision making on hospital bed allocation; (viii) evaluation of the system created using real hospital data and with the help of professionals responsible for the bed allocation in a hospital.

1.4 Thesis Outline

This thesis proposal is organised as follows. Chapter 2 presents the background related to multi-agent systems, explainable artificial intelligence and hybrid intelligence, argumentation theory, dialogue systems, JaCaMo framework, ontology, runtime verification and runtime monitoring language, as well as bed allocation domain. Chapter 3 shows our proposed approach. Chapter 4 describes the developed multi-agent intentional dialogue system framework called MAIDS and its modules. In Chapter 5, we present the case study in the hospital bed allocation domain. Chapter 6 presents the related work. Last, Chapter 7 final considerations and discusses future work.

Chapter 2

Background

In this chapter, we bring the background related to this thesis. We will discuss the following topics: Multi-agent systems, Explainable artificial intelligence, Argumentation theory, Dialogue systems, JaCaMo framework, Ontology, Runtime verification and Hospital bed allocation domain.

2.1 Multi-agent Systems

Multi-agent systems are systems composed of multiple agents. They seem to be a natural metaphor for building and understanding a wide range of artificial social systems and can be applied in several different domains [Woo09]. There are two interlocking strands of work in multi-agent systems: the one that concerns individual agents and the one that deals with the collections of these agents. In practice, agents rarely act alone. They usually inhabit an environment that contains other agents. Each agent can control, or partially control, the environment, which is called the "sphere of influence". It may happen that these spheres of influence overlap with what causes the environment to be controlled jointly by more than one agent. In this case, to achieve the desired result, an agent must also consider how other agents may act. These agents will have some knowledge, possibly incomplete, about the other agents [BHW07a].

Wooldridge [Woo09] believes that to be able to understand a multi-agent domain is essential to understand the type of interaction that occurs between agents. For intelligent autonomous agents, the ability to reach agreements is extremely necessary, and for this, negotiation and argumentation skills are often necessary. In [FGM03], the authors cite two types of studies in multi-agent systems; the first one is agentcentred multi-agent systems (ACMAS), which study, at the level of an agent, states and the relationship between those states and their general behaviour, which are projected in terms of the agent's states of mind. The second one is organisation-centred multi-agent systems (OCMAS), which are systems whose foundations reside in the concepts of organisations, groups, communities, roles, and functions, among others. An OCMAS is not considered in terms of mental states but in capacities and constraints, which are considered organisational concepts, such as functions, tasks, groups, and interaction protocols.

In a multi-agent system, the organisation is the collection of roles, relationships, and other social structures that govern agents' behaviour. Every multi-agent system has some form of organisation, even if it is implicit and informal. Organisations guide the mode of interaction between agents, which may influence data flows, resource allocation, authority relationships, and various other features of the system [HL04].

2.2 Explainable Artificial Intelligence and Hybrid Intelligence

Explainable Artificial Intelligence (XAI) is a research field that refers "to make AI systems results more understandable to humans" [AB18]. These results must be clear (in non-technical terms) and provide explanations about decisions made [DDE20]. The need for explaining a decision/reasoning/action was discussed as early as the 1970s. It started with the expert systems development and the need for those systems to explain their decisions not only with traces but also with justifications [ABdR+20]. Recommend systems also become facilitators to produce a justification to help users decide whether to follow a recommendation. However, many artificial intelligence (AI) applications face difficulties explaining their autonomous decision and actions to human users [GSC+19a]. For instance, when the application involves the logic of axioms and inferences, the reasoning process's explanation can be complex [DDE20]. Indeed, if the system is limited to showing the violated axioms, then the user will probably not understand the results [DDE20].

In [ANCF19], the authors argue that systems which heavily adopt AI techniques are increasing, and making them explainable is a priority. Thus the area of explainable artificial intelligence emerged intending to foster transparency and trustworthiness. Conducting a systematic review of the literature in goal-driven XAI (explainable agency for robots and agents), they conclude that: (i) most approaches are based on simple scenarios; (ii) almost all approaches focus on robots/agents explaining their behaviours to the human users (very few concern about inter-agent explainability); and (iii) only a few works addressed the issues of customisation and context-awareness.

Explainable AI has two main research directions: data-driven XAI (explaining black-box algorithms) and goal-driven XAI (explainable agency) [ANCF19]. While **data-driven domains** focus on the concept of interpretability (i.e., their operation can be understood by a human), **explainable agency** reflects autonomous agents explaining their actions and the reasons leading to their decisions. In our work, we have used goal-driven XAI to make suggestions and support decisions from experts.

Authors believe that automatically generated explanations have a fundamental mechanism to increase user trust in systems [YJ95]. The explanations can also help users make better decisions or persuade them to make one particular choice [TM11]. In [Mil19], the author describes that while there are many ways to increase trust and transparency of intelligent agents, two complementary approaches are considered the most relevant: (i) generating decisions in which one of the criteria taken into account during the computation is how well a human could understand the decisions in the given context, which is often called *interpretability* or *explainability*; and (ii) explicitly explaining decisions to people, i.e., *explanation*. Also, explanations can be partial or complete. Partial models reveal the important parts of their reasoning. In contrast, the complete models provide complete and transparent explanations [GSC+19a].

Regarding evaluating and measuring whether an XAI system is more understandable by the user than another XAI system, it is an open challenge [GSC+19a]. Usually, subjective measures are used, such as user satisfaction when using the system. It also includes evaluation frameworks and argumentation [GSC+19a]. For instance, in [WYAL19], they proposed a theory-driven, user-centric XAI framework to provide pathways to mitigate reasoning failures due to cognitive biases.

Regarding Hybrid Intelligence, it is defined by Akata *et al.* [ABdR+20] as "the combination of human and machine intelligence, augmenting human intellect and capabilities instead of replacing them and achieving goals that were unreachable by either humans or machines." The HI is underlined by the interaction between AI agents and humans, considering human intentions and expertise, as well as ethical, legal, and societal issues. The capability to explain motivations, actions, and results is an essential element in this interaction. And we must keep in mind that humans are used to an environment where norms and values (often implicitly) trace which goals and actions are permitted or even desirable.

2.3 Argumentation Theory

For reasoning with inconsistent information based on the construction and comparison of arguments, we have argumentation as a promising approach [RA06]. Argument-based techniques are used to facilitate interaction between rational agents and to specify the reasoning of autonomous agents. We might see argumentation as the interaction based on principles of different arguments which are potentially conflicting and have the objective of reaching a consistent conclusion [MPR06].

The ultimate aim of argumentation is to resolve a potentially conflicting point of view that may be subject to both justification or refutation depending on the available information, arriving at a consistent conclusion. Argumentation can also be used for theoretical reasoning, which refers to what to believe, and practical reasoning, which refers to what to do. This second requires capturing arguments about non-propositional attitudes such as goals and desires [MPR06, RA06].

In [MPR06], the authors show two main research lines of argumentation in multi-agent systems. First, autonomous agent reasoning, such as decision making under uncertainty and belief revision. Second, since argumentation naturally provides tools for designing, implementing and analysing advanced forms of interaction between rational agents, it might be seen as a vehicle for facilitating multi-agent interaction. Thus, there are two main types of problems encountered in the multi-agent systems addressed by argumentation [MPR06]:

- Formation and revision of beliefs and decisions: based on incomplete, conflicting or uncertain information, argumentation provides a systematic means to resolve conflicts between arguments and to form beliefs and decisions.
- Rational interaction: Argumentation provides a framework to ensure that dialogue between participants, which have conflicting points of view, adheres to certain principles.

The argumentation process can be considered as a kind of reasoning about arguments made to determine the most acceptable of them. It is necessary to be able to evaluate the reasons why a fact is valid by combining and comparing arguments to arrive at a conclusion [RA06]. All forms of reasoning with incomplete information assume that a defeasible statement can be believed only if there is no evidence to the contrary [Dun95a].

Argumentation-based reasoning determines when an assertion can be believed, provided that the arguments supporting this assertion are successfully defended against the counter-arguments of attack. Hence, we can affirm that an agent's internal arguments that support its beliefs and the external arguments that support contradictory beliefs characterise the beliefs of a rational agent [Dun95a].

In [PB16], an argument structure developed based on an agent-oriented programming language is formally defined and implemented. In addition, it describes an argument-based reasoning mechanism that allows agents to construct and define the acceptability of arguments, where conclusions are drawn on a provisional basis, and it is possible to invalidate them when new information comes. This structure presented in [PB16] allows the development of multi-agent applications that use argumentative techniques for making decisions, as well as allows richer dialogues through the exchange of arguments.

2.3.1 Abstract Argumentation

An essential component of human intelligence is argumentation. The human being's ability to engage in arguments is essential for understanding new problems, carrying out scientific reasoning, and defending opinions in everyday life [Dun95a]. In order to understand how human beings, even illiterate, carry out their reasoning, the abstract argumentation approach was proposed. Following the proposal by Baroni *et al.* [BTV20], we observe that, in addition to being a formalism to capture various approaches to common sense reasoning, abstract argumentation is also "a methodology for providing abstractions of problems along three dimensions: arguments, attacks and acceptability, the latter amounting to what agents have to accept given what they know (i.e. the arguments and associated attacks)."

Building arguments is something that human beings learn from childhood. The structure of arguments can be problematic in general, and different models of argument use different representations of the structure of arguments. They use reasons, rules, assumptions, and logical deductions in various combinations.

Without disagreement, there is no arguing. Because of that, the notion of attack is essential in abstract argumentation. Depending on the model of argument used, the types of attacks included may vary.Taking into account only the attack relationship, we can assess the acceptance of an argument. Evaluating acceptance without considering any other underlying details, we guarantee uniform applicability to a wide variety of situations. Intuitively, we can accept arguments if they can be defended against attacks. A set that does not contain arguments that attack other arguments in the set or even themselves and that manages to counter all attacks of arguments in the set is said to be an admissible set of arguments.

There are several special types of admissible sets in abstract argumentation. Some of the main ones are: (1) Stable extensions in which the arguments in that set do not attack each other but attack arguments outside the set - there are not always stable extensions, but sometimes there are several possible ones; (2) Preferred extensions (credulous semantics) in which the maximum admissible set is sought - there is always at least one preferred extension, and there may also be several; (3) Grounded extensions (skeptical semantics) in which the set has acceptable and unambiguous arguments - there is always precisely one [Dun95a].

2.3.2 Monological and Dialogical Argumentation

In the 1980s, argumentation emerges in the area of AI as a powerful method to represent a diverse range of knowledge and to support various types of reasoning [Ton14]. A meaningful way to explore argumentation in systems is to use dialogues, i.e., dialogical argumentation, especially between agents. For agents to engage in dialogue, there must be some protocol that both parties will follow to make sense of the information exchanges [BCD07].

In addition, a single agent can also use argumentation techniques to execute their individual reasoning because they need to make decisions about complex preference policies in a highly dynamic environment. According to [BH08], this type of argument is called monological because it involves a single agent or entity that classifies knowledge to build arguments for and against a given conclusion. When an agent has the ability to argue, he increases his autonomy and provides him with more intelligent behaviour [Pru11].

Monological and dialogical argumentation are closely related, because a rational agent will execute monological argumentation to build and evaluate the acceptability of its arguments, and later also consider other arguments communicated during a dialogue, to understand which arguments have won and what is the result of the dialogue. Such understanding might depend on the type of dialogue the agent is participating, for example, the seven basic types of dialogues described by Walton [Wal10].

In that paper, Walton defines each dialogue model by its initial situation, the participants' individual goals, and the aim of the dialogue as a whole. He defines the six basic types of dialogue as [WK95]: inquiry, negotiation, information-seeking, deliberation, and eristic dialogue. And more Discovery dialogue as [MP01a]. Table 2.1 shows the definition of these seven basic types of dialogue.

The goal of each party in the persuasion dialogue is to win over the other side. To this end, both parties seek to find arguments that defeat the other's thesis or put it in doubt. It is essential in a persuasive dialogue that participants agree with the

Table 2.1: Seven basic types of dialogue [Wal10]

Type of Dialogue	**Initial Situation**	**Participant's Goal**	**Goal of Dialogue**
Persuasion	Conflict of Opinions	Persuade Other Party	Resolve or Clarify Issue
Inquiry	Need to Have Proof	Find and Verify Evidence	Prove (Disprove) Hypothesis
Discovery	Need to Find an Explanation of Facts	Find and Defend a Suitable Hypothesis	Choose Best Hypothesis for Testing
Negotiation	Conflict of Interests	Get What You Most Want	Reasonable Settlement Both Can Live With
Information-Seeking	Need Information	Acquire or Give Information	Exchange Information
Deliberation	Dilemma or Practical Choice	Co-ordinate Goals and Actions	Decide Best Available Course of Action
Eristic	Personal Conflict	Verbally Hit Out at Opponent	Reveal Deeper Basis of Conflict

issue to be discussed in the opening phase [Wal10]. This type of dialogue always arises from a conflict of opinions and aims to resolve that conflict so that, in the end, a stable agreement is reached [WK95]. While persuasion dialogue is highly adversarial, the inquiry is cooperative in nature. Its goal is to prove that a statement is true or false, or, if none of these options can be proved, to prove that there is not enough evidence to prove the truth or falsity of the statement. This type of dialogue seeks to draw conclusions only based on premises that can be firmly accepted as true or false [Wal10]. "Inquire, as a type of dialogue, is like persuasion dialogue (and unlike deliberation) in that it aims at a stable agreement. However, it resembles deliberation (and differs from persuasion dialogue) in that it arises from a problem rather than conflict: something is not known definitely to be true or false" [WK95].

In a discovery dialogue, we want to discover something not previously known [MP01a]. The question of whether the truth should be determined only arises during the course of the dialogue itself, unlike the inquire dialogue, for example, in which the proposition that must be proven true is designated before the course of argument in the dialogue [Wal10]. Discovery dialogues, then, have a similar feature to deliberation dialogues, since the course of action adopted by the participants can also emerge in the course of the dialogue itself. However, these two types of dialogue differ since, in a deliberation dialogue, the participants' preferences or emotions can play an essential role in selecting an ideal course of action, while in a discovery dialogue, the participants are also seeking truth, but, there may be many possible truths and they might filter the truths they discover by what is interesting, novel or important [MP01a].

Despite arising from a conflict as occurs in the persuasion dialogues, negotiation, like deliberation, has the objective of reaching a decision as the basis for action, "both deliberation and negotiation are inherently practical types of dialogue, geared to action to enable practical affairs of life and human commerce to go ahead" [WK95]. Walton [Wal10] states that deliberation is a type of collaborative dialogue. In this type of dialogue, the parties' actions are collectively directed towards a common goal. The parties agree on a proposal that can solve a problem that affects them all, taking all of their interests into account. On the other hand, unlike negotiation and deliberation, the information-seeking type of dialogue aims to correct or eliminate an asymmetric distribution of information between the parties. In the case of information-seeking, knowledge already exists. It only needs to be communicated from one party to the other. Finally, the eristic dialogue starts from a conflict as the initial situation but is more modest in its aspirations than any other of the main types [WK95]. In eristic dialogues, the parties attack each other's arguments, without necessarily seeking the truth.

2.3.3 Argumentation Schemes

Argumentation schemes are argument/reasoning patterns found in daily conversation, and in specific argumentation, such as scientific argumentation (scientific reports, discourses, etc.) [WRM08]. Argumentation schemes provide an elegant way to represent and analyse these typical argument patterns that are naturally found in the reasoning construction.

According to Panisson [Pan19], "Argumentation schemes are considered deductive and inductive forms of argument, added the so-called defeasible, presumption or abductive part". An argument that is considered defeasible can be strong enough to provide evidence to guarantee its rational acceptance of its conclusion, even if it is not strong in itself. This happens when its premises are acceptable [Tou58]. We can provisionally accept the conclusion of a revocable argument concerning known evidence, but it may be necessary to retract it as new evidence emerges.

Argumentation schemes can also be considered an attempt to reach a conclusion when there is a situation of uncertainty or lack of knowledge. The most important types of schemes are revocable by nature [WRM08]. As new evidence comes into consideration, arguments that had previously been accepted can be defeated. This is what we call the revocability factor, and it brings us to the problem of how the schemes are linked rationally.

Revocability is usually linked to a dialogue, where a proponent, based on an argumentation scheme, asserts some conclusion and the opponent, also based on the argumentation scheme, can ask a critical question that needs to be answered successfully by the proponent [WRM08]. The results of possible critical questions can have problematic conclusions in several aspects. To evaluate a given argument in a particular case concerning a context of dialogue in which that argument occurred, the argumentation scheme critical questions are used together. The critical questions aim to raise doubts about the structural link between the premise and the conclusion. Associated critical questions are used to judge the strength or weakness of an argument based on an argumentation scheme. In this way, we can judge whether the argument is good or fallacious. During a dialogue, when an opponent receives an argument, he/she can: (1) ask a critical question related to that argument; (2) provide an argument against the claim of the received argument; (3) challenge one of the premises of this argument; or (4) accept the conclusion of this argument as a commitment [Pan19].

For example, the *Argument from role to know in MAS* (*role to know* for short) from [PMB21, PB20] 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)**".

The associated critical questions are:

- **CQ1** Does playing role R imply knowing whether A holds?
- **CQ2** Is ag an honest (trustworthy, reliable) source?
- **CQ3** Did ag assert that A is true (or false)?
- **CQ4** Is ag playing role R?

To allow agents to instantiate arguments from argumentation schemes, Panisson and colleagues [PB16, PB17c, PMB21, PB20] 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]:

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

[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.

where the agents can instantiate such argumentation schemes with the information available to them and evaluate the acceptability of the conclusion based on the interactions among such instantiated arguments [PMB21, PB20].

The critical questions are represented in Jason as beliefs as follows:

```
CQ1: role_to_know(Role,Conclusion)[as(role_to_know)]
CQ2: reliable(Agent)[as(role_to_know)]
CQ3: asserts(Agent,Conclusion)[as(role_to_know)]
CQ4: role(Agent,Role)[as(role_to_know)]
```

The argumentation-based reasoning mechanism links the critical question and the inference rule for a particular argumentation scheme using the annotation [as(as_name)], with as_name the name of the argumentation scheme. Note that there will be no critical questions for many argumentation schemes.

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 ↦ john, Role ↦ doctor, Domain ↦ cancer, Conclusion ↦ causes(smoking, cancer)}.

Further, argumentation schemes combined with natural language templates can be used for translating arguments from a computational representation to a natural language representation [PEB21, FPE+22]. For example, the natural language template for the argumentation scheme role_to_know is as follows:

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

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

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

A variety of problems in artificial intelligence can be addressed using argumentation schemes. Even if we disregard the physical aspects of interaction with the world and consider only artificial agents' reasoning capacities, there are still significant challenges for these agents, such as uncertainty and incompleteness, that we can consider as two fundamental problems to be addressed [WRM08]. And in general, in addition to interacting with the world, these systems of reasoning also need to interact with humans, which needs to be understood dialectically.

2.4 Dialogue Systems

A dialogue system is a computer program capable of communicating with a user using natural language [ABS13]. This communication is carried out through text or voice. Dialogue systems can be categorised into two types: (I) goal-driven dialogue systems (or task-oriented systems), and (II) non-goal-driven systems [CLYT17, WY16]. A typical goal-driven dialogue system extracts the necessary information from the user's utterances to achieve a goal, for example, flight booking. While the non-driven dialogue system, user input is answered without any specific goal [WY16]. A Chatbot, also known as chatterbot [TJ20], is a non-goal-driven dialogue system example. Virtual assistants, such as Cortana, Google Assistant, and Siri, have embedded chatbots.

A good dialogue system should mix the two categories [WY16]. For instance, a virtual assistant chats with the user in a non-goal-driven way, but there are also dialogues that are goal-driven such as writing an email. In recent years, famous companies such as Google (Dialogflow[2]), IBM (Watson[3]), Microsoft (Luis[4]) have invested in creating platform for the development of chatbots. These Platforms have mechanisms for natural language processing (NLP) and dialogue management. In addition, they also have integration with services and applications.

Intents are the main component used by these platforms. According to the user's input (what the user says), the corresponding intent is mapped to provide the appropriate response (or action). In other words, it has the ability to understand what the user says, and it can choose or generate a response which can be based on the current input and the context of the conversations [RAMI17]. The developer is responsible for registering each intention, and it can call external services that were developed. An intention can have a large set of user inputs; this is because a human can speak the same thing in many ways. Thus, each intention must have alterna-

[2]https://dialogflow.com/
[3]https://www.ibm.com/watson/br-pt/
[4]https://www.luis.ai/home

tive syntactic forms to express the same semantics. In DialogFlow, for instance, it is possible to use a prebuild agent with some intentions, and the developer can complement it according to the domain chosen.

Due to advances in NLP, we have seen a growing demand for integrating speech and text recognition capabilities with interactive software applications [VGBM17]. This integration can minimise one of the most important challenges in the field of human-computer interaction, which is maintaining and enhancing the willingness of the user to interact with the technical system [NM17]. Just as people use natural language for human communication, people want to use their language to communicate with computers [SA07].

2.5 The JaCaMo Framework

JaCaMo is a framework that allows the multi-agent-oriented programming. This framework consists of the integration of three existing platforms: Jason - for programming autonomous agents, CArtAgO - for programming environmental artefacts and Moise - for programming multi-agent organisations [BBH+13]. A multi-agent system programmed in JaCaMo has Jason agents that are organised and follow roles according to a hierarchical structure programmed in Moise. These agents work in environments based on distributed artefacts programmed using CArtAgO. Figure 2.1 shows the JaCaMo dimensions overview.

Jason (Agent dimension) is an agent-oriented programming language that is an interpreter for the AgentSpeak [BHW07b] language. Agents programmed in Jason use the BDI model. The main idea of this model is to model the process of deciding which action to take to achieve certain objectives [RG+95].

Moise is related to the organisation dimension, where agents can be part of groups and follow specific roles [HSB07, BBH+13]. Also, with Moise, the schemes are defined, that is, the structure of goals of the organisation that is decomposed into sub-goals, which are grouped into missions. This organisation is programmed in an XML file.

CArtAgO, the last dimension, is related to a simulation of the environment where is defined the artefacts [RPVO09]. These artefacts define the environment's structure and behaviour, representing all resources that agents need in the simulation. Agents can discover, create, and use artefacts at runtime [BBH+13]. Artefacts are programmed in Java language.

The combination of these dimensions provides us with a complete framework for

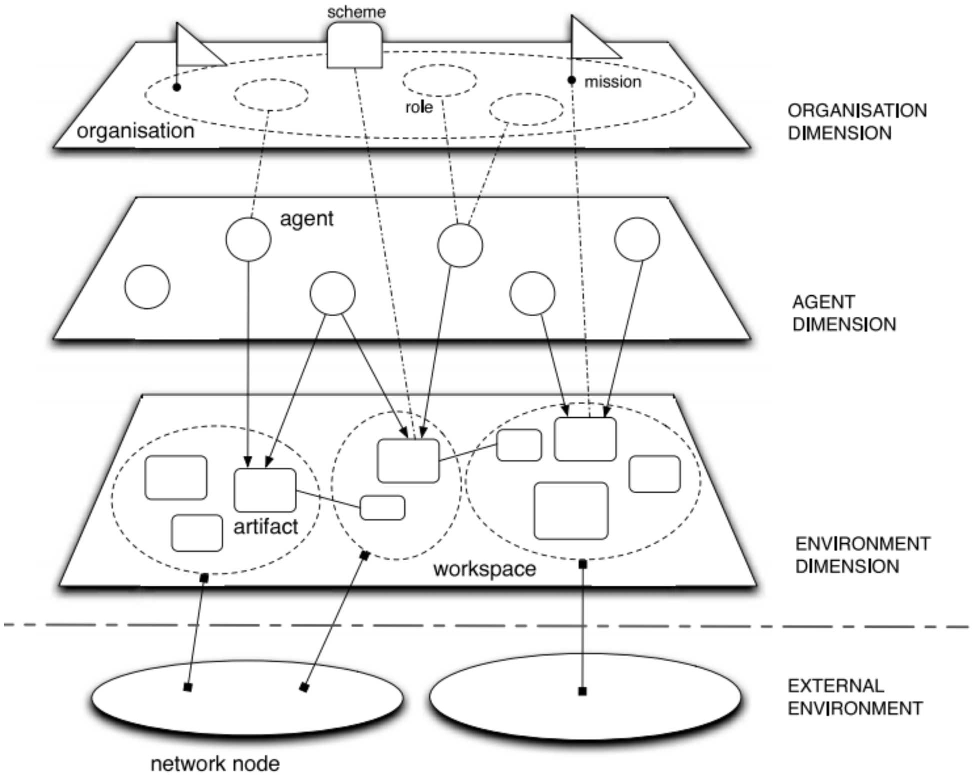

Figure 2.1: Overview of the three dimensions of JaCaMo [BBH+13]

developing multi-agent systems with agents, organisations, and environments. Although this framework is relatively recent, several researchers have already explored its potential such as in [CR20, AH19]. In [CR20], the authors describe the JaCa-Android approach, a framework based on the JaCaMo that allows for designing and programming smart mobile apps. While [AH19] presents jacamo-web, an interactive programming IDE for developing Multi-Agent Systems. Also, JaCaMo's full potential was used in a worldwide multi-agent programming contest (MAPC) that took place in 2018. Our team used the framework and came in second place [KCD+18]. There was no first place that year.

2.6 Ontology

The term "Ontology" was born on Philosophy and refers to the study of existence, taking into account "what" and "how" things exist in the world following a hierarchical classification [Gru95]. For AI, it describes a domain knowledge that follows basic principles, such as identifying domain classes, the hierarchy of classes, properties, and their relationships to reflect reality [VASS05]. Ontology is also a framework that supports the process of modelling a domain to provide a collection of terms and their semantic interpretation [BHHK03]. Also, it can have rules that are called Axioms that constrain the interpretation and well-formed use of these terms [Gru95].

Several formal languages have been created over the years to represent ontologies. The most popular ontology languages are KIF, OWL, RDF + RDF(S), and DAML+OIL [KV11]. These languages have web-based standards, which means that they process web information. However, the standard language, 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 [VASS05].

Some tools help in the construction of ontologies; one of the most popular is Protégé [GMF+03]. Protégé[5] allows the use of plugins for ontology graphic visualisation and semantic reasoning. It also has formats available for ontology upload and download, such as RDF/XML, Turtle, OWL/XML, and OBO. When developing an ontology, Semantic Web Rule Language (SWRL) [OMD09] can be used to model more sophisticated inferences. They are specified in the following format: $\texttt{pre}_1, \ldots, \texttt{pre}_\texttt{n} -> \texttt{conc}$, with $\texttt{pre}_1, \ldots, \texttt{pre}_\texttt{n}$ the n premises of the rule, and conc the conclusion of the rule.

In this work, we are interested in using Ontology together with MAS. This is motivated because Ontologies can help in agent knowledge modelling and reasoning, task representation, and inference [TL08, SBMV15]. We found some approaches that integrated ontology information with agent-oriented programming languages. For example, in [MVBH05a], they created AgentSpeak-DL that extends agents' belief base with description logic where agents can share knowledge by using ontology languages such as OWL. In [KB08a], the authors used AgentSpeak-DL to develop JASDL (Jason AgentSpeak–DescriptionLogic), which provides agents' ontology manipulation capabilities using the OWL API. JASDL allows plan trigger generalisation based on ontological knowledge and using such knowledge in querying the belief base.

Also, in [MABR11], the authors created CooL-AgentSpeak, an extension of AgentSpeak-DL with plan exchange and ontology services. CooL-AgentSpeak uses a CArtAgO artefact as an ontology repository tool to store a set of ontologies and provides ontology matching/alignment. Another approach is presented in [FPH+15], whose authors developed a CArtAgO artefact to give access to ontological information. The artefact uses OWL API to create, manipulate, query, and serialise ontologies coded

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

in OWL. Thus, agents can use operations such as loading the ontology, adding instances, and adding concepts.

2.7 Runtime Verification and Runtime Monitoring Language

Runtime Verification (RV) [BFFR18] is a kind of formal verification technique that focuses on checking the behaviour of software and hardware systems. It dynamically checks whether the event traces generated by single runs of a system under scrutiny comply with the formal specification of its expected correct behaviour [LS09]. Concerning other formal verification techniques, such as Theorem Provers [Lov78] and Model Checking [Cla97], RV is considered more dynamic and lightweight. This is mainly due to its focus on checking how the system behaves while it is currently running, which is important from a complexity perspective. Furthermore, RV analyses what the system produces (*i.e.*, everything that can be observed in the system); in other words, it does not need to simulate the system to check all possible execution scenarios.

The system under scrutiny and the specification of the properties to be verified are used as the inputs of the RV process (Figure 2.2). The specification denotes sets of event traces where a trace is called *valid* according to the specification if and only if it belongs to such a set. This specification is usually defined in either a Domain-Specific Language (DSL) or a programming language.

A monitor synthesised from the specification is responsible for consuming the observed events generated by the system, emitting verdicts and, sometimes, some feedbacks [AFFM21]. The system can use these feedbacks for error recovery when monitoring continues after deployment. In turn, a formal property is a formal representation of how we expect the system should behave. The monitor's job is to verify whether such a property holds at runtime.

The monitor uses a 4-valued logic [BLS07] where the two conclusive verdicts *False* and *True* are combined with the two inconclusive ones *MaybeFalse* and *MaybeTrue*. This is needed because the verdicts emitted after each observed event may be inconclusive [LS09] as the monitor only inspects a finite prefix of the possibly infinite trace generated by the system execution.

Monitors are suitable for checking properties that need continuous monitoring since they are usually deployed in conjunction with the system under analysis. This is especially true in critical security scenarios, where a system failure can cause injury,

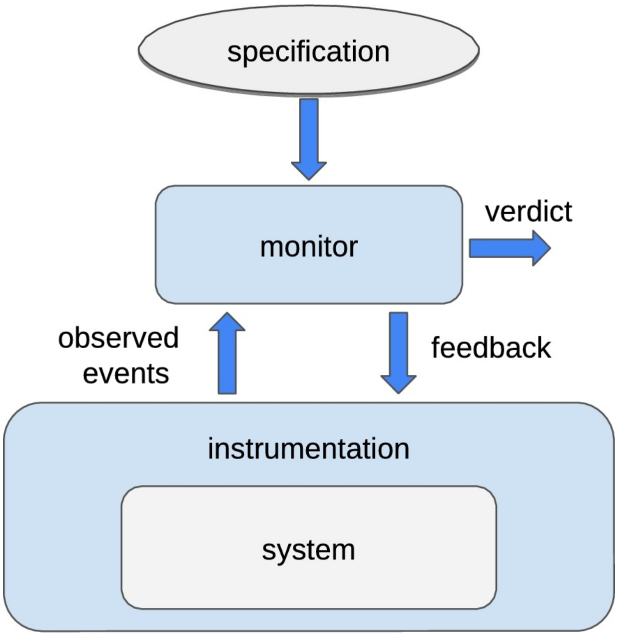

Figure 2.2: Overview of the RV process [AFFM21]

loss of money, and even death. The protocols involved in the communication between agents and human beings can be very complex and hard to track. Moreover, agents are usually mainly focused on the reasoning and reactive aspects, while the consistency of the protocols is given for granted. However, above all, in the case of human beings in the loop, such an assumption cannot be made. Therefore, RV is a suitable candidate to keep track of the protocols to check whether the current agents' enactment is consistent (or not) with the expected protocol. Furthermore, such consistency checking is crucial in safety-critical scenarios, such as healthcare, where a protocol violation can be costly.

Runtime Monitoring Language[6] (RML [AFFM21]), in turn, is a Domain-Specific Language (DSL) for specifying highly expressive properties in RV (such as non-context-free ones). We use RML in this paper for its support of parametric specifications and its native use for defining interaction protocols. In fact, the low-level language, on which RML is based upon, was born for specifying communication protocols [AFM16, AFM17].

Since RML is just a means for our purposes in this thesis, we only provide a simplified and abstract view of its syntax and semantics. However, the complete presentation can be found in [AFFM21].

[6] `https://rmlatdibris.github.io/`

In RML, a property is expressed as a tuple $\langle t, ETs\rangle$, with t a term and $ETs = \{ET_1, \ldots, ET_n\}$ a set of event types. An event type ET is represented as a set of pairs $\{k_1 : v_1, \ldots, k_n : v_n\}$, where each pair identifies a specific piece of information (k_i) and its value (v_i). An event Ev is denoted as a set of pairs $\{k'_1 : v'_1, \ldots, k'_m : v'_m\}$. Given an event type ET, an event Ev matches ET if $ET \subseteq Ev$, which means $\forall(k_i : v_i) \in ET \cdot \exists(k_j : v_j) \in Ev \cdot k_i = k_j \land v_i = v_j$. In other words, an event type ET specifies the requirements that an event Ev has to satisfy to be considered valid.

An RML term t, with t_1, t_2 and t' as other RML terms, can be:

- ET, denoting a set of singleton traces containing the events Ev s.t. $ET \subseteq Ev$;
- $t_1\ \ t_2$, denoting the sequential composition of two sets of traces;
- $t_1 \mid t_2$, denoting the unordered composition of two sets of traces (also called shuffle or interleaving);
- $t_1 \land t_2$, denoting the intersection of two sets of traces;
- $t_1 \lor t_2$, denoting the union of two sets of traces;
- $\{let\ x;\ t'\}$, denoting the set of traces t' where the variable x can be used (i.e., the variable x can appear in event types in t', and can be unified with values).
- $t'*$, denoting the set of chains of concatenations of traces in t'

Given an event type ET, the term $\neg ET$ denotes its negation. Specifically, $\forall_{Ev}.ET \subseteq Ev \iff \neg ET \not\subseteq Ev$. Other RML terms can also be negated. For example, if the term is $ET_1 \land ET_2$, its negation is $\neg ET_1 \lor \neg ET_2$; the same reasoning can be applied to the remaining operators.

Event types can contain variables. For example, $ET(ag1, ag2) = \{sender : ag1, receiver : ag2\}$, where we do not force any specific value for the sender (resp., receiver) of a message (in this case, the events of interest would be messages). This event type matches all events containing the sender and receiver. When an event matches an event type with variables, such as in this case, the variables get the values from the event. For instance, if the event observed would be $Ev = \{sender : \text{“}Alice\text{”}, receiver : \text{“}Bob\text{”}\}$, it would match ET by unifying its variables as follows: $ag1 = \text{“}Alice\text{”}$, and $ag2 = \text{“}Bob\text{”}$. This aspect is important because we can use variables in RML terms to enforce a specific order of messages. For instance, in this very high-level example, we could say that when a message from $ag1$ to $ag2$ is observed, the only possible consequent message can be a message from $ag2$ to $ag1$. Since the first event has unified the two variables, the second event will have to be a message from Bob to $Alice$ (otherwise, this would be considered a violation). Naturally, this is only the intuition behind it, but it should help to grasp the expressiveness of RML and how variables can be exploited at the protocol level to enforce specific orders amongst the messages.

2.8 Hospital Bed Allocation Domain

Resource management in hospitals aims to maximise resource usage and avoid hospital overcrowding. In the last decades, healthcare systems have been facing a massive increase in demand, which has led to an ongoing need to improve and optimise operational processes and quality control methods [EW22]. In addition, hospital managers have studied ways to improve the use of hospital resources and maintain high occupancy rates without creating chaos in the emergency room or long queues [GdCR+18b]. The demands on hospitals and the growing financial constraints make planning and efficient allocation of hospital beds increasingly difficult [MR11].

Brazil had the second highest burden of coronavirus disease 2019 (COVID-19) worldwide. More than 36.55 cases and 0.93 deaths per thousand inhabitants as of December 31, 2020. In addition, as of October 1, 2021, the country has recorded the highest number in the world (402,220) of deaths caused by COVID-19 [BMT+22]. With a considerable number of COVID-19 patients worldwide during the pandemic, the hospitals faced massive shortages of isolation beds with an appropriate environment to prevent airborne microorganisms from entering corridors, which could result in secondary infections. On the other hand, they still had to consider the needs of non-COVID-19 patients. Even during the pandemic, many non-COVID-19 patients, mainly those in emergency cases, still require hospitalisation. Therefore a critical management problem faced was how to optimally allocate the limited amount of hospital beds between COVID-19 and non-COVID-19 patients [MZG22].

Hospital beds are scarce, and therefore, allocating them optimally plays an essential role in the overall planning of hospital resources [TEDF+12]. Availability of beds in specialised wards for each patient's medical condition can reduce errors and improve the quality of patient care [ZEJ+22]. However, when performing an efficient bed allocation, it is necessary to consider many variables that make it difficult for a human to work out the best solutions without any assistance. Also, this is a complex task computationally, so artificial intelligence incorporated into multiagent systems can be helpful in this context. Effective management of such resources has always been a challenge for managers, given that hospital settings are highly dynamic and uncertain. Uncertainty in this domain comes from the fact that hospitals need to accommodate patients undergoing elective (scheduled) and emergencies requiring multiple specialities in a wide range of departments with varying constraints [GdCR+18b] as well as handling emergency cases that are impossible to predict. This makes bed management an essential part of planning and controlling operational capacity and an activity involving the efficient use of resources [PGB03]. Thus, it would be interesting to have a system that assists in suggesting better bed allocations for the professional responsible for this task.

Chapter 3

Conversational Agents Using Structured Dialogues

3.1 Overview of the Approach

This approach supports the development of dialogue systems based on BDI agents to assist humans in decision making. The use of dialogues in natural language facilitates the interaction and adaptation of human users. Also, the use of argumentation theory and ontology can make dialogues more useful for them. The argumentation ability provides more autonomy and smarter behaviour to agents. Ontology, on the other hand, helps agents to organise domain knowledge, as it contains all relevant entities and their relationships, providing the possibility of ontological reasoning about the domain. We have made our approach adaptable enough to be applied across multiple domains.

We have created a framework for developing dialogue systems built on top of Jason.In that framework, agents have three separate components of their belief base: (i) argumentation schemes for the application domain that the dialogue system is aimed for, following a structured (rather than abstract) argumentation approach; (ii) an OWL ontology about that same domain; and (iii) a theory of mind component storing presumed mental attitudes of other agents. With that multi-part belief base setting, our framework provides support for agents having a structured dialogue where the main line of argumentation is based on the argumentation schemes knowledge component. Still, it can lead to subdialogues when ontological or ToM issues must be resolved. We have focused on the expressiveness of dialogue systems where agents have such a multi-part belief base and the ability to engage in such structured dialogues.

The idea of subdialogues is in line with general ideas on nested dialogues (see, e.g., [BH07]), but we give a practical protocol limiting such "digressions", thus avoiding unnecessary computational burden. In fact, the multi-part belief base accompanied by the dialogue structure with subdialogues has a clear impact on efficiency, given that commitment stores of subdialogues can be deleted when they are completed. Importantly, because this is all in the context of an agent-oriented programming language that is formally based on the BDI architecture, we have precise and computationally-grounded [Woo00] semantics for the mental attitudes that agents have and ascribe to others.

Although all the knowledge of the multi-part belief base, if suitably translated from the various sources, could be merged and used by argumentation systems as a single knowledge base, there are two main advantages of the modular approach we propose here: (i) it allows us to reuse existing ontologies on top of the more expressive (argumentation-based) reasoning that we may want to program for particular systems (i.e., encouraging reusability of existing ontologies in agent development); and (ii) it allows the agent strategy to "consciously" decide when to move on to an ontological argumentation[1] or argumentation about other agents' mental attitudes before returning to the main line of argumentation.

3.2 A Multi-Part Belief Base

In our system, agents have a belief base with at least three main components. Each of these is based on work appearing in the literature, in particular: (i) defeasible and strict rules based on an AgentSpeak implementation of d-Prolog [Nut93]; (ii) the CooL-AgentSpeak language which allows for the use of ontologies and ontology alignment; and (iii) recent work on the theory of mind for AgentSpeak agents. The subsections below describe each of these separate bases forming our *multi-part belief base*.

3.2.1 The CooL-AgentSpeak Language

CooL-AgentSpeak stands for "Cooperative description-Logic AgentSpeak" [MAB+14]. It resulted from various strands of past work on combining AgentSpeak with onto-

[1]It should be noted that the *ontological argumentation* term we introduce here bears no relation to what in Philosophy is known as "ontological argument". We use this expression to refer specifically to multi-agent dialogues based on argumentation theory, where the content of the arguments being exchanged make explicit reference to a formal ontology.

logical reasoning [MVBH05b, KB08b, AMHB04], and has the following features:

- it extends the AgentSpeak programming language with *ontological knowledge*, formally through a description logic and in practical implementation through the use of OWL ontologies;
- it has an explicit *cooperation strategy* to be used when agents exchange plans;
- it takes advantage of *ontology matching* functions so that agents using different ontologies can communicate, in practice using available ontology matching services.

Because it has all these features that are, in practice, important in multi-agent settings, we take that programming language as the basis for this component of the belief base that we require for our structured dialogue approach with ontological argumentation.

However, for this purpose, we only need the belief base component of that language (Ont), so we do not show the other components of a CooL-AgentSpeak agent (ag); the excerpt of the syntax that is of interest here is summarised in Figure 3.1.

$$
\begin{array}{lll}
\mathrm{ag} & ::= & \mathrm{Ont\ ps\ cs\ ms} \\
 & & \\
\mathrm{Ont} & ::= & \mathrm{ABox\ TBox} \\
\mathrm{ABox} & ::= & \mathrm{at_1} \ldots \mathrm{at_n} \quad (\mathrm{n} \geq 0) \\
\mathrm{TBox} & ::= & \mathrm{C_1} \equiv \mathrm{D_1} \ldots \mathrm{C_n} \equiv \mathrm{D_n} \quad (\mathrm{n} \geq 0) \mid \\
 & & \mathrm{C_1} \sqsubseteq \mathrm{D_1} \ldots \mathrm{C_n} \sqsubseteq \mathrm{D_n} \quad (\mathrm{n} \geq 0) \mid \\
 & & \mathrm{R_1} \equiv \mathrm{S_1} \ldots \mathrm{R_n} \equiv \mathrm{S_n} \quad (\mathrm{n} \geq 0) \mid \\
 & & \mathrm{R_1} \sqsubseteq \mathrm{S_1} \ldots \mathrm{R_n} \sqsubseteq \mathrm{S_n} \quad (\mathrm{n} \geq 0) \\
\mathrm{C, D} & ::= & \mathrm{A} \mid \neg\, \mathrm{C} \mid \mathrm{C} \sqcap \mathrm{D} \mid \mathrm{C} \sqcup \mathrm{D} \mid \forall \mathrm{R.C} \mid \exists \mathrm{R.C} \\
\mathrm{R, S} & ::= & \mathrm{P} \mid \mathrm{R} \sqcap \mathrm{S} \mid \mathrm{R} \sqcup \mathrm{S} \\
\mathrm{at} & ::= & \mathrm{C(t)[\texttt{o}(oid),\ \texttt{src}(bsrc)]} \mid \\
 & & \mathrm{R(t_1,\ t_2)[\texttt{o}(oid),\ \texttt{src}(bsrc)]} \\
\mathrm{bsrc} & ::= & \texttt{self} \mid \mathrm{aid_1, \ldots, aid_n} \mid \texttt{percept} \\
\mathrm{oid} & ::= & \textit{a string identifying an ontology} \mid \texttt{self} \\
\mathrm{aid} & ::= & \textit{a string identifying an agent}
\end{array}
$$

Figure 3.1: Excerpt of the CooL-AgentSpeak Syntax

Following [MVBH05b], CooL-AgentSpeak used $\mathcal{ALC}$ as the underlying description logic [BN03]. The definition of classes and properties belonging to the ABox of the ontology assumes the existence of identifiers for primitive (i.e., not defined) classes and properties (metavariables A and P, respectively). New classes and properties can be defined using constructs such as $\sqcap$ and $\sqcup$ that represent the intersection and the union of two entities, respectively. The TBox is a set of axioms establishing equivalence and subsumption relations between classes and between properties. With respect to [BN03] and [MVBH05b], the syntax was extended to allow *annotations* (as available in Jason) of concepts and properties. Furthermore, in practice, we use OWL ontologies, so OWL inference rules [HPSB$^+$04] can also be used.

An agent belief is an atom belonging to the ABox annotated with o(oid), where *oid* is the identifier of the ontology. We use *oid=self* for "naive beliefs" [KB08b], i.e., a normal AgentSpeak belief that does not relate to an ontology. Along the lines of [VMWB07], beliefs are also annotated with sources src(*bsrc*), where *bsrc* can be either an agent identifier *aid* specifying the agent which previously communicated that information or *self* to denote beliefs created by the agent itself, or *percept* to indicate that the belief was acquired through the perception of the environment.

3.2.2 Argumentation-based Reasoning in Agent Programming

Our agents have an internal rule-based argumentation mechanism capable of generating (evolving) arguments. Rule-based argumentation frameworks can be found in the literature, for example, in [Pra11], which extends well-known work by Dung [Dun95b], with structures to arguments based on *strict* and *defeasible* rules, and the work of Berariu [Ber14] and [PMVB14] which extend Jason agents with such argumentative reasoning capabilities. In this work, we use specifically the approach and implementation by Panisson et al. [PMVB14], which has been extended to consider the representation of argumentation schemes (i.e., reasoning patterns) for various applications domains [PB17b].

Agents in our framework should use an argumentation semantics that allows a unique set of acceptable arguments such as *grounded semantics* defined in [Dun95b] and used in [Ber14, Pra11], or the *defeasible semantics* defined in [GMAB04] and used in [PMVB14]. Furthermore, agents only accept propositions/claims which they do not have an acceptable argument against (i.e., the *cautious* attitude), and agents only assert propositions/claims for which they have an acceptable argument (i.e., the *thoughtful* attitude) [PM03, PWA02].

In our dialogue approach, we need to determine the acceptability of an argument from the agent's perspective (i.e., whether the agent does or does not have an argument for a given claim). That implementation referred to above and upon which we have built this component of our belief base provides that for us.

3.2.3 Theory of Mind in Agent Programming

For an agent to interact successfully, effectively achieving its goals in a multi-agent environment, it is essential for it to be able to model and reason about other agents' minds. The term theory of mind is used to refer to such an ability. ToM has been much discussed in the context of multi-agent systems, as it can be noted in [dWVV12, DWV11], which investigated the advantages of using different levels of ToM in games played by agents, and [BA11, HDM^{+}12, HSM^{+}13, ON09, RTO13], which investigated the advantages of modelling the opponent when considering strategies in argumentation-based dialogues.

More recently, works have been published on how ToM can be modelled in Agent-Oriented Programming Languages (AOPL) [PSM^{+}18a, SPB^{+}18]. The authors discuss how AOPLs are suitable as a basis for analysing social interaction, including, for example, when intelligent agents can be dishonest [PSM^{+}18b]. Pioneering work relating ToM to AOPLs appeared in [KC04], building on seminal work on the semantics of speech-acts [CP79].

In this work, we take advantage of existing approaches to ToM in agent programming in order to model and reason about other agents' mental attitudes. Similar to ontological inquiries, in our approach, agents' ToM may also be the target of subdialogues, in which agents will argue about their own or other agents' mental attitudes. In fact, ToM subdialogues may be more often required than ontological ones, given how susceptible ToM is to be incorrect or incomplete. Even with probabilistic models, such as in [SPB^{+}18], when an agent builds a model of other agents' minds, this model is often different from reality, given that many factors can mislead the perception of the mental attitudes of others, and given that agents change their mental attitudes constantly, particularly in highly-dynamic multi-agent systems.

3.3 The Basis for Argumentation-Based Dialogues

In our mechanism, agents argue using a subset of the speech acts found in the literature of argumentation-based dialogue [AMP00, PM03, PWA02]. This work builds on an existing approach by Panisson *et al.* [PMVB15a]. It will be summarised in this section to make the full operation of our approach clearer. The particular performative verbs used here and their informal meaning are as follows:

assert: an agent that performs an assert utterance declares, to all participants of the dialogue, that it is committed to defending this claim — the receivers of the message become aware of this commitment;

accept: an agent that performs an accept utterance declares, to all participants of the dialogue, that it accepts the previous claim (assert) of another agent — the receivers of the message become aware of this acceptance;

question: an agent that performs a question utterance desires to know the reasons for a previous claim of another agent or, in case of an information-seeking dialogue, desires to know if the receiver can provide the information requested in the content of a *question* message.

challenge: the receiver of the message, who previously committed to defend a claim, should now provide the support set for that claim;

justify: the justify message is similar to the assert message but is used as a response to a challenge message previously received, whereby the agent provides the support to its previous claim.

We adopt the formal definition of the semantics of these speech acts from the work by Panisson *et al.* [PMF$^+$14, PMVB15b] which specify precisely the effect of the speech acts in the agent's mental state, as well as in the multi-agent dialogue as a whole. The formal semantics allows for direct implementation of the effects of receiving and sending the speech-act in a BDI-based agent-oriented programming language based on the mental attitudes used in that specification. From that work, we use the stated effects of each speech act on an agent's commitment store (CS) for the specification of our protocol, as described below.

The CS consists of one or more structures, accessible to all agents in a dialogue, containing commitments made by the agents during the dialogue[2]. The CS is simply a subset of the knowledge base, and the union of the CSs can be viewed as the global state of the dialogue at a given time [PWA02].

In the course of the dialogue, the agents use rules that define how the CS is updated. Such rules are part of the semantics used in this work and are summarised as follows (making reference to the agent that *uttered* the speech act):

- *assert*: the agent's CS is updated with the asserted content p: $CS \leftarrow CS \cup \{p\}$;
- *accept*: the agent's CS is updated with the accepted content p: $CS \leftarrow CS \cup \{p\}$;
- *question*: no effect on the CS;
- *challenge*: no effect on the CS; and

[2]Other names are used for CS, such as *dialogue obligation store* in [MP05] and *dialogue store* in [STT01].

- *justify*: the agent's CS is updated with the justified content contained in the set of rules and facts S (the support for a challenged claim p): $CS \leftarrow CS \cup S$;

Note that in our implementation, we support multi-agent interaction, so messages can be directed to a particular agent or to '$*$', which is used to denote all agents taking part in a particular dialogue. A message has the format `performative(sender, receiver, content)`. Besides the performative verbs used in individual messages, a *dialogue game protocol* restricts the moves allowed to agents. The dialogue game restricts the moves but, as usual in such mechanisms, it also determines the alternative moves available to agents at any point in the multi-agent interaction. In fact, an interesting approach to determine an agent's individual strategy to participate in such interaction is through planning, as done, for example, in [BCH17, PFF$^+$14].

The particular dialogue game approach we use in this work is built upon fundamental ideas that appeared in [PWA02, PM03]. That work formalises the preconditions (called "rationality rules") for an agent to make each type of dialogue move and what commitment store updates ensue. Furthermore, that work shows how those moves can be used to build dialogues for various purposes among those mentioned in Section 2.3.2 depending on the agent's intention. Our case study in Chapter 5 shows in practice the sort of dialogue that the implementation of such rationality rules supports. Note that they provide the means for agents to engage in a dialogue, but our case study further shows when an agent chooses to move to an ontological subdialogue, following the rules we formally introduce in the next section.

3.4 Multi-Agent Dialogues with Underlying Ontological and ToM Arguments

We present the structure of subdialogues, which can be seen in Figure 3.2. Agents engage in a dialogue about some subject (a claim put forward by the agent initiating the main dialogue). The dialogue proceeds normally following a particular protocol and using the knowledge base δ. In the case study reported here, for example, we use a multi-agent version of the dialogue protocol referred to in Section 3.3 for both the main dialogue and each of the two types of subdialogues. What we formalise later in this section is precisely when an agent may digress from the mainline of argumentation and move on to an ontological or ToM subdialogue. As seen in Figure 3.2, after some moves in either type of subdialogue, the agents involved in the dialogue must go back to discussing the main subject; that is, the main line of argumentation is suspended when a subdialogue starts, and it is only resumed when that subdialogue finishes.

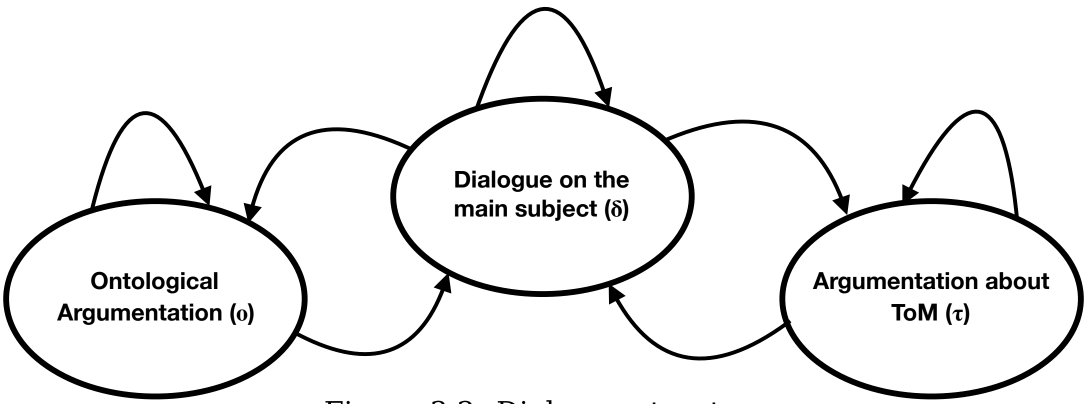

Figure 3.2: Dialogue structure

The move towards a subdialogue is best explained by an example. Suppose we have $P(c)$ as a strict fact, $P(c) \wedge D(c) \rightarrow Q(c)$ as a defeasible rule, $C(c)$ in the ABox, and $C \sqsubseteq D$ in the TBox of the o belief-base component. If, after asserting $Q(c)$ the agent is questioned about $D(c)$, the justification involves the ontological assertions. When presented with them, the other agent might disagree that $C(c)$ or disagree with the TBox statement if the ontologies are not correctly aligned. After that dialogue phase (i.e., a subdialogue) is finished, the main dialogue flow resumes. The result of the subdialogue, of course, will affect the main line of discussion (the dialogue about a particular subject of interest in the domain following the available defeasible knowledge). The agents may conclude the subdialogue by unanimously agreeing that $D(c)$, that $\neg D(c)$, or finishing the subdialogue inconclusively. In the latter case, the main dialogue will continue so that agents try to reach an agreement on the main subject despite being unable to agree on the ontological issue.

Similarly, we might have a subdialogue to further inquire about ToM assumptions, in which case the subdialogue uses knowledge from the ToM component. Support for ToM in our framework is done by incorporating the work on ToM for agent programming languages discussed in Section 3.2.3. Yet, those beliefs are particularly susceptible to being incorrect and incomplete. This is partly because of the intrinsic benevolence assumption in the rules for generating ToM, but also because, in a dynamic environment, the agent's mental attitudes can change rapidly without further communication exchange that would have allowed the ToM to be updated. Again, after a ToM subdialogue, the result will affect the main dialogue in the same ways mentioned above.

Our work includes the formalisation of a novel dialogue-subdialogue structure, and we use an existing protocol for the (sub)dialogues. Besides implementing the rules that support the dialogue protocol, our framework requires derivation of conclusions to be obtained for each of the 3 belief-base components when the agent needs to respond to a `challenge` message. For the defeasible component, the existing argumentation-scheme-based implementation already produces an AgentSpeak list with the sequence of rules used to derive a particular conclusion. For the ontology component, we automatically translate the semantic rules presented in the ontologies to a computational representation of argumentation schemes (defeasible rules), allowing the agents to reason using the defeasible component. Finally, for the ToM component, the justification directly references the rules of the operational semantics that govern how ToM is updated in an agent language [PSM+18a] that we incorporated into our system.

3.4.1 Formalisation of Participating Agents

As seen in the previous section, our work builds on two other separate pieces of work in the literature: domain-specific strict and defeasible rules and facts, and one or more ontologies (without lack of generality, we use only one in the formalisation), and a ToM (i.e., the information about other agents' state of mind that is kept updated through communication); note that the all messages exchanged by agents may contribute to ToM updating, including the messages exchanged following the overall dialogue protocol we present in this section and the associated protocol governing (sub)dialogues. An agent participating in dialogues created with our framework is formalised as follows.

Definition 1 (Agent) *An agent that takes part in our structured-dialogue argumentation protocol is defined as a tuple $\langle \delta, o, \tau, \pi, \varepsilon, \iota \rangle$, where δ is a set of defeasible and strict rules and facts (in the AgentSpeak style based on d-Prolog); o is a CooL-AgentSpeak style ontology-based belief base; τ is an AgentSpeak representation for ToM following the approach described in the previous section; π is the set of plans to achieve goals forming the agent's know-how (i.e., its plan library); ε is a set of AgentSpeak events which include, for example, recent goal adoptions (i.e., goals that are not yet intentions); and ι is the agent's current set of intentions (partially executed, partially instantiated plans to achieve goals).*

Note that (δ, o) are two now separate components replacing what would normally be simply one set of beliefs representing the agent's current belief base. We use C_i to refer to component C of agent i. Introducing further notation, we say that an agent can build an acceptable argument S that supports a claim p (denoted as $S \models p$) from one of its knowledge bases and the commitment store of the other participants. For example, agent i can build an acceptable argument S, which supports a conclusion p, from its defeasible knowledge base (δ_i) and the commitment store of j (CS_j) (denoted $(\delta_i \cup CS_j) \models S$).

3.4.2 Subdialogue Rules

We now introduce the rules governing the high-level dialogue structure; that is, the rules that allow agents to initiate the subdialogue we would like them to have in our framework. They should be interpreted in the context of normal *dialogue rules* [MP02, MP01b] determining a protocol that governs the interactions between the agents, given their strategies whereby each agent moves by performing one of the utterances allowed by the protocol. Such rules, effectively determining a dialogue game [MP02], are often expressed as if-then rules, which are then easy to implement.

The dialogue rules specify the moves that each player can make, and so specify the *protocol* under which the dialogue takes place [AMP00]. As mentioned before, the permitted moves in each (sub)dialogue follow the existing protocol discussed in Section 3.3. Instead of the usual if-then rules, we use a different style, similar to operational semantic rules, to formalise new performatives that are required to support the dialogue structure. In order to do so formally, we first define the overall dialogue setting.

Definition 2 (Subdialogue Game) *A subdialogue game is formally represented as a tuple $\langle MD, SD_1, \ldots, SD_n, MS, DR \rangle$, where MD is the main dialogue, SD_i ($1 \leq i \leq n$) are n possible subdialogues, MS is a finite set of allowed moves between any of the dialogues, and DR a set of dialogue rules governing the moves between the various (sub)dialogues. Our model assumes that digressing to a subdialogue suspends the dialogue on the main subject, which is only resumed when the subdialogue finishes.*

We propose one particular subdialogue game as follows.

Definition 3 (Ontological-ToM Subdialogue Game) *An Ontological-ToM subdialogue game, denoted by SDG^{OT}, is formally defined by $\langle MD^{OT}, SD^{O}, SD^{T}, MS^{OT}, DR^{OT} \rangle$.*

Arguments can be formed from the commitment store of the main dialogue and the knowledge in δ of each agent. The SD^O subdialogue uses o plus its commitment store and SD^T uses τ and another particular commitment store as well as ε and ι (so that the agent may refer to its own desires and intentions, as well as beliefs[3]). The formalisation of the two other components is given below in this section. First, we formalise a particular running instance of dialogue following our Ontological-ToM Subdialogue Game.

[3] For a formalisation of the BDI modalities for AgentSpeak agents, see [BFVW06].

Definition 4 (Dialogue Instance) *A particular dialogue instance following our Ontological-ToM Subdialogue Game is defined as* $\langle dID, \mathcal{A}, SDG^{OT} \rangle$ *where* dID *is a unique dialogue instance ID,* $\mathcal{A}$ *is the set of agents (in this paper we assume the same set of agents participates in the main as well as all subdialogues), and* SDG^{OT} *is as per Definition 3.*

Definition 5 (Dialogue Moves) *We denote a move in* MS^{OT} *as* $v(i, j, \varphi)$*, where* v *is the performative verb used for that move, made by agent* i*, addressed to agent* j*, regarding content* φ*. We consider the following set of performatives, denoted by* P *(see Section 3.3):* `assert`*,* `accept`*,* `question`*,* `challenge`*,* `justify`*,* `closedialogue`*,* `ontoargsubdlg`*,* `tomsubdlg`*,* `closesubdlg`*, and* `failsubdlg`*. The content of a move (*φ*) can be an argument (a set of formulæ) or just a formula (e.g., in an* `assert` *move, the content is a formula and in a* `justify` *move, the content will be a support set for a claim made in a previous* `assert` *move).*

The dialogue rules in DR^{OT} indicate the possible moves that an agent can make following a previous move by another agent. They are presented here in the form of an inference rule in a similar presentation style as used in operational semantics of programming languages, except that here the conclusion part of the rule state which dialogue move (or *transition*) is allowed when the premises of the rule hold. A dialogue transition $l \longrightarrow r$ means making the r move in response to a previously received message l. When necessary to make that clear, a move r may be written r_M, r_O, or r_T depending on whether it took place in the main, ontological, or ToM (sub)dialogue. In the premises, existential quantification is assumed, and horizontal space between formulæ denotes conjunction. When multiple rules can fire, those are precisely the points where an individual agent strategy will determine how the dialogue unfolds (and as mentioned before, planning is one possible technique to help determine optimal dialogue strategies). We use $*$ to denote messages that are not directed towards a particular agent but to all agents taking part in the dialogue. The specific rules DR^{OT} that govern our subdialogue structure are as follows.

$$\frac{f \in \delta_j \qquad C(t) \in f \qquad o \vdash C(t)}{\texttt{challenge}(i, j, f)_M \longrightarrow \texttt{ontoargsubdlg}(j, *, C(t))_O} \tag{OAsdlg1}$$

$$\frac{f \in \delta_j \qquad R(t_1, t_2) \in f \qquad o \vdash R(t_1, t_2)}{\texttt{challenge}(i, j, f)_M \longrightarrow \texttt{ontoargsubdlg}(j, *, R(t_1, t_2))_O} \tag{OAsdlg2}$$

Rule OAsdlg1 says that if an agent challenges, in the main dialogue *M*, a formula in which $C(t)$ appears, and C is related to an ontology class, we can enter a subdialogue to discuss whether t indeed is an instance of class C. Rule OAsdlg2 is exactly like OAsdlg1 but for an ontology relation $R(t_1, t_2)$ rather than a class. Note that it is assumed in the formalisation, without loss of generality, that the participating agents have only one ontology, which they have individually aligned using CooL-AgentSpeak. In practice, a `ontoargsubdlg` message could include a parameter for the URI of the particular OWL ontology referred to by the agent starting the subdialogue. When agents receive an `ontoargsubdlg` message, they know they have to switch their moves to a fresh instance of the subdialogue protocol.

$$\frac{\forall a \in \mathcal{A}. o_a \models \varphi}{\texttt{closedialogue}(i, *, \varphi)_O \longrightarrow \texttt{closesubdlg}(i, *, \varphi)_M} \qquad \text{(CloseOAsdlg1)}$$

$$\frac{\forall a \in \mathcal{A}. o_a \models \neg\varphi}{\texttt{closedialogue}(i, *, \neg\varphi)_O \longrightarrow \texttt{closesubdlg}(i, *, \neg\varphi)_M} \qquad \text{(CloseOAsdlg2)}$$

$$\frac{\begin{array}{cc} a \in \mathcal{A} & o_a \models \varphi \\ b \in \mathcal{A} & o_b \models \neg\varphi \end{array}}{\texttt{closedialogue}(i, *, \varphi)_O \longrightarrow \texttt{failsubdlg}(i, *, \varphi)_M} \qquad \text{(FailAOsdlg)}$$

Rule CloseOAsdlg1 states that when the `closedialogue` performative is used by one of the agents to finish a dialogue which was an ontological subdialogue, that leads to the closing of the subdialogue with success (`closesubdlg`), in case all agents agreed on φ, and thereafter to the resuming of the main dialogue. Note that although we specify the condition from the point of view of the belief base of the participating agents, that can also be checked from the commitment stores of the subdialogue. Rule CloseOAsdlg2 is exactly like CloseOAsdlg1 except that it applies when all agents accept $\neg\varphi$ instead.

It should also be noted that following a $\texttt{closesubdlg}(i, *, \varphi)$ message, the commitment store of the main dialogue is updated with the fact that now all agents accept φ (i.e., they reach an agreement about whether that ontological issue holds or not). When instead rule FailAOsdlg applies, the main dialogue is resumed with no alteration in the commitment stores. The dialogue will have to continue despite the disagreement on φ.

The closing rules for ToM subdialogues are very similar, so for our purposes here, we only need to formalise the rules for starting a ToM subdialogue.

$$\frac{f \in \delta_j \qquad \mathsf{Mod}_{a \in \mathcal{A}}(\varphi) \in f \qquad \tau_j \vdash \mathsf{Mod}_{a \in \mathcal{A}}(\varphi)}{\texttt{challenge}(i, j, f)_M \longrightarrow \texttt{tomsubdlg}(j, *, \mathsf{Mod}_{a \in \mathcal{A}}(\varphi))_T} \qquad \text{(OTsdlg)}$$

where $\mathsf{Mod} \in \{\mathsf{Bel}, \mathsf{Des}, \mathsf{Int}\}$. Rule OTsdlg says that if a formula f is challenged by an agent and that formula involves a subformula which is associated with the ToM component of the belief base, we may start a subdialogue to discuss specifically whether the mental attitude of a particular agent does in fact hold, i.e., there is a divergence between their ToMs.

Definition 6 (Divergence between agents' ToM) *Considering two agents $i, j \in \mathcal{A}$, there is divergence between their ToM about some mental attitude $\mathsf{Mod}_k(\varphi)$, for some agent $k \in \mathcal{A}$, when $\tau_i \models \mathsf{Mod}_k(\varphi)$ and $\tau_j \not\models \mathsf{Mod}_k(\varphi)$.*

We assume that agents have a consistent ToM about their own mental attitudes (they have perfect introspection about their own mental attitudes), i.e., $\forall\varphi \in \{\delta_i \cup \iota_i\}$ then $\mathsf{Mod}_i(\varphi) \in \tau_i$. Also, they have a consistent ToM about other agents, i.e., $\mathsf{Mod}_j(\varphi)$ and $\mathsf{Mod}_j(\neg\varphi)$ does not hold in τ_i simultaneously. Thus, we have the following scenarios for ToM subdialogues: (i) When $j = k$, i.e., agent i has a divergent model about j's mental attitude $\mathsf{Mod}_j(\varphi)$, agent j can inform its current mental attitude $\mathsf{Mod}_j(\varphi)$ to i. (ii) When $i = k$, i.e., agent j has a divergent model about i's mental attitude $\mathsf{Mod}_i(\varphi)$, agent i can inform its current mental attitude $\mathsf{Mod}_i(\varphi)$ to j. (iii) When $j \neq k$ and $i \neq k$, i.e., agents i and j have a divergence about another agent k's mental attitude $\mathsf{Mod}_k(\varphi)$, agents i and j may argue about the current mental attitude $\mathsf{Mod}_k(\varphi)$ of k.

When the mental attitude causing a divergence between two agents' ToM refers to a belief (i.e., $\mathsf{Mod}_i(\varphi) = \mathsf{Bel}_i(\varphi)$), ToM subdialogues will be characterised as an *information-seeking (sub)dialogue* for cases (i) and (ii) above and an *inquiry (sub)dialogue* for case (iii). When the mental attitude causing such divergence refers to a desire or intention, (e.g., $\mathsf{Mod}_i(\varphi) = \mathsf{Des}_i(\varphi)$), those ToM subdialogues may result in more complex interactions, possibly involving persuasion in case (iii). Such subdialogues about agents' desires/intentions are supported by carefully designed argumentation schemes introduced by D.Walton [Wal19].

FINAL REMARKS

This chapter presented an overview of our approach. We show the components used in our approach to assisting users in decision making. Also, it presents the definitions and rules used in our dialogue protocol. We have idealised our approach in a way that allows us to explain to the user how our agents arrived at certain conclusions. We worked with a more semantic representation and representations of the mental state of the dialogue participants, allowing the formulation of coherent justifications expressed in natural language, therefore, easy for human participants to understand. This fact has revealed the potential of our approach to be used in the practical development of explainable intelligent systems as well as systems supporting hybrid intelligence. The contents of this chapter have been published in [EPV$^+$].

Chapter 4

Multi-Agent Intentional Dialogue System

4.1 Overall Architecture

The architecture of the Multi-Agent Intentional Dialogue System (MAIDS) framework we developed in this work is shown in Figure 4.1. As it is shown, our framework relies on the use of `Dial4JaCa` (see Section 4.2) as an interface to dialogue platforms such as Dialogflow[1] or Rasa[2]. The `Human users` can interact with the chatbot through voice or text. This interaction is classified into intents by Dialogflow or Rasa and sent to `Dial4JaCa`, which makes the request available to the `Communication agent` assigned to that specific user.

One or more `Communication agents` can be instantiated, each one responsible for representing a `Human user`. This allows us to customise the responses given to the user based on a previously defined (or learned) profile. With this profile, the application avoids giving too many explanatory answers to a user with a specialist background and avoids giving very superficial answers to users without a specialist background since it translates the responses of the `Assistant` (the result of the MAS reasoning) into natural language messages, using templates as described in [PEB21, FPE$^+$22], to be sent to its corresponding `Human user`. Furthermore, the ability to instantiate multiple communication agents, one for each system's user, also allows an `Assistant` agent to engage in multiparty conversations, helping a team or a group of users make joint decisions.

The `Assistant` agent is responsible for communicating with other agents in search of information and performing argumentation reasoning. Besides multiple communication agents,

[1] https://cloud.google.com/dialogflow/es/docs
[2] https://rasa.com/docs/

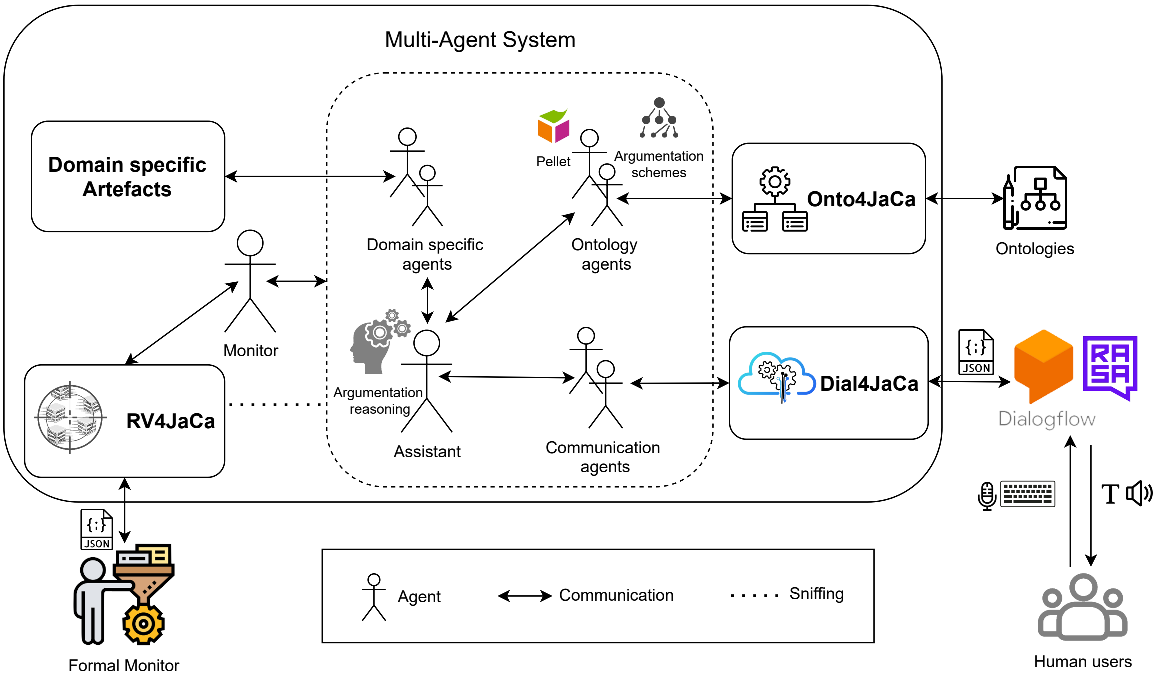

Figure 4.1: MAIDS Architecture

several `Ontology agents` can be instantiated in MAIDS, allowing the MAS to consult several ontologies simultaneously through `Onto4JaCa` (see Section 4.3). These agents can also perform ontological reasoning using the Pellet reasoner [SPG$^+$07] and its open-source continuation effort Openllet[3]. In addition, these agents can translate OWL inference rules [HPSB$^+$04] automatically to defeasible rules (representing argumentation schemes) and use them during the reasoning process.

In order to address the specificity of different application domains, `Domain specific agents` are added to the system, and they can query different `Domain specific Artefacts` depending on the system's needs. Also, MAIDS uses the `RV4JaCa` (see Section 4.4) to perform runtime verification. It observes all messages exchanged between agents and sends them through a REST (Representational State Transfer) request to a `Formal Monitor` where the properties that need to be checked are defined. The `Formal Monitor` processes the received information, checking whether any property has been violated. In case of violation, RV4JaCa adds information about the failure in the `Monitor` agent's belief base that warns the agents involved in the exchange of messages that there has been a violation. This makes it possible for our agents to take action to recover from the failure that the breach caused.

Note that MAIDS is a modular framework, which means that its components can be used

[3]https://github.com/Galigator/openllet

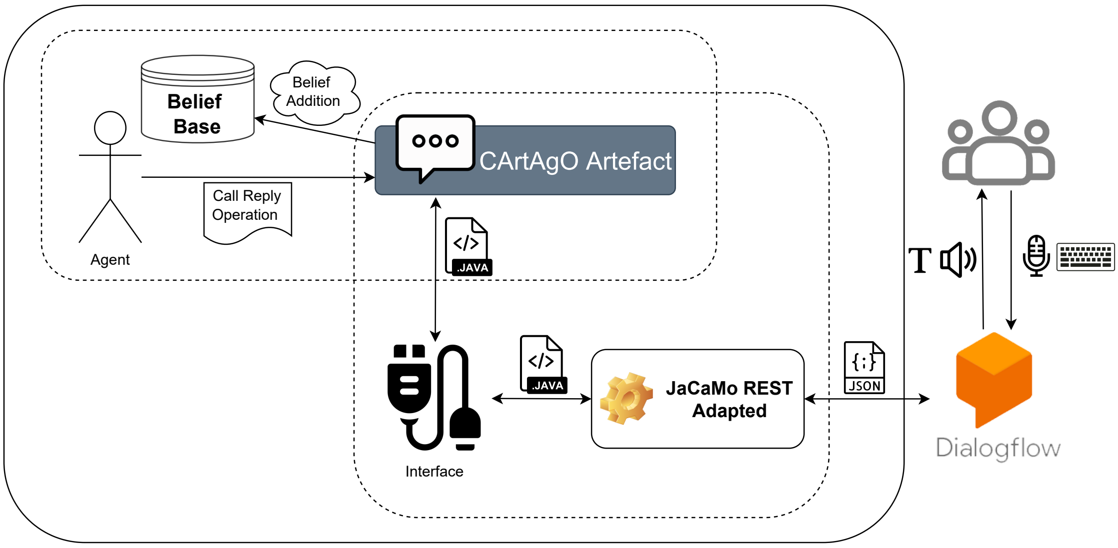

Figure 4.2: Dial4JaCa Architecture

together or separately, depending on the needs of the multi-agent system. The following sections explain each of the modules in more detail.

4.2 Dial4JaCa

Dial4JaCa[4] [EDK+21b, EDK+21a] integrates the JaCaMo framework with Dialogflow and, therefore, allows us to implement intelligent agents that can communicate with humans through natural-language interaction. We have made this integration sufficiently adaptive to be applied to different applications and domains. To do so, we use modular components which can be imported into any multi-agent system developed in JaCaMo. Figure 4.2 depicts an overview of the Dial4JaCa architecture.

To develop the bridge that links a MAS application, developed using JaCaMo, to a natural language processing platform, e.g., Dialogflow, we use part of the open-source project `JaCaMo REST`[5] [AHK20]. `JaCaMo REST` allows a MAS to interact with services or applications on the web and to be managed and updated by other applications. In our approach, Dial4JaCa receives requests from `Dialogflow` through `JaCaMo REST`. It conveys the received information to a `CArtAgO Artefact` responsible for making the requests available to the agents. The data are received in JSON (JavaScript Object Notation) format and immediately transformed into a Java object, which becomes available in a `CArtAgO Integration Artefact`.

[4] `https://github.com/smart-pucrs/Dial4JaCa`
[5] `https://github.com/jacamo-lang/jacamo-rest`

```
+request(ResponseId, IntentName, Params, Contexts) :true
<-  .print("Request received ",IntentName," of Dialog");
    !reply(ResponseId, IntentName, Params, Contexts).
```

Listing 1: An example of how an agent perceives a request.

This `Interface` allows the integration artefact to perceive whenever a request is received. Upon realising the arrival of a requisition, the artefact performs a *belief addition* in the *belief base* of all agents who are observing it (i.e., the observable properties). That belief contains all relevant information about the request. Doing so, the agents that *focus* on that artefact are able to decide whether they are going to react to such requests or not.

Regarding `Dialogflow`, it uses the `fulfillment`[6] service to communicate to external APIs. We use this service to integrate Dialogflow with the MAS, passing requests through the resource-oriented abstraction layer from `JaCaMo REST`. With the resulting communication interface, intelligent agents developed in JaCaMo perceive not only information about an intent triggered by the user's speech, but also parameters and contexts that Dialogflow has collected in each interaction.

Contexts[7] are another important concept in Dialogflow. They are similar to contexts in natural language conversations. That is, it is a relationship between the text and the situation in which it occurs. To process a user's expression in natural language, Dialogflow can use the context to correctly match it with an intent. By doing so, it is possible to control the flow of a conversation. In addition, intents can also have parameters[8] that are values extracted from the user's expression.

Listing 1 shows an example of a plan in Jason that agents can use to react to a belief addition (`+request`). It informs the agent that a new request from the user has arrived. In this simple example, the agent creates a new goal to `!reply()`, which will result in a sequence of instructions to be carried out.

In addition to the intention's name, Dial4JaCa also allows the agent to have access to `contexts` and `parameters`, which are captured by Dialogflow. This information is recorded in its belief base and might be used during its reasoning cycle. `Contexts` is a list, as shown in the Listing 2, where for each element of this list: `Name` corresponds to the context name; `LifeSpanCount` corresponds to the context lifespan; and `[param(Key, Value), param(Key1, Value1)]` matches the context parameter list. Even if there is no context, receiving a list of parameters is possible. That list has the same structure as the parameter list in the context, where `Key` corresponds to the parameter's key, and `Value` corresponds to

[6] `https://cloud.google.com/dialogflow/es/docs/fulfillment-overview`

[7] `https://cloud.google.com/dialogflow/es/docs/contexts-overview?hl=en`

[8] `https://cloud.google.com/dialogflow/es/docs/intents-overview?hl=en`

```
context(Name,LifeSpanCount,[param(Key,Value),param(Key1,Value1)])
```

Listing 2: An example of context.

```
+!reply(ResponseId, IntentName, Params, Contexts)
  : (IntentName == "Reply With Context")
<- .print("The context will be created next.");
    contextBuilder(ResponseId, "test context", "1", Context);
   .print("Context created: ", Context);
    replyWithContext("Hello, I am your Jason agent, and I am
                            responding with context", Context).
```

Listing 3: An example of a plan to reply to a request with a context.

the value recorded by Dialogflow.

Dial4JaCa also allows an agent to send new contexts to Dialogflow along with the response to a request. An operation named `contextBuilder` is available for this purpose. This operation receives as a parameter the response *id* (`responseId`), the name of the context to be created (`contextName`), and the lifespan of the context (`lifespanCount`). It returns a context formatted according to the Dial4JaCa specification. To reply with a context, the agent calls the operation `replyWithContext`, which takes as a parameter a string with the text that the chatbot must tell the user and the context created by the `contextBuilder`. We exemplify this process in Listing 3.

Dial4JaCa also provides the `reply` operation in case an agent does not need to send out a context and the `replyWithEvent` operation that allows an agent to send events to Dialogflow.

One should be particularly careful when dealing with timeouts in Dialogflow. They indicate that the conversation flow should continue, and a reply from the MAS is no longer expected. This behaviour can be triggered, for instance, when an agent calls upon external services to reason about an appropriate response. Upon realising that an agent takes too long to respond, Dial4JaCa automatically fires an event in Dialogflow, containing the same name as the current remote intent (getting rid of the blank spaces). Doing so allows the agent more time to finish its reasoning. This process is perceived by neither the user nor the agent. Dialogflow allows these sorts of events to be fired up to three times in a row. In our tests, this mechanism has provided sufficient time for an agent to deliver a response with no endless waiting time on the user side.

In our implementation, we have approached only the Dialogflow platform. However, preliminary investigations have shown that with small changes in our code, we can also use other natural language processing platforms such as Watson, or Luis. It was confirmed by a project called Rasa4JaCa [Cus22], which has extended our approach to be able to use JaCaMo agents with the open-source natural language processing platform Rasa[9].

In addition, some works have used Dial4JaCa to integrate chatbots and multi-agent systems in different domains. In [dSCVMB21], the authors developed a Chatbot using a multi-agent organisation to support collaborative learning. In [OEK+22] a multi-agent architecture used as a basis for an ambient assisted living system to assist visually-impaired and elderly people was presented. Furthermore, in [GM22], it was introduced VEsNA, a framework for managing virtual environments via natural language agents.

Also, in [Esf23], they propose a tool called Onto2Conv, which is responsible for reading a customised domain-specific ontology and generating the files to feed the information for the chatbot and the MAS. It feeds them with their acceptable skeleton and format to make these chatbots robust and their development faster and less error-prone when using Dial4JaCa. Furthermore, they have implemented donnaMAMi, a motivational agent in the mirror in the domain of psychology and domestic violence against women, which also uses Dial4JaCa as a bridge between Dialogflow and the multi-agent system developed using JaCaMo. In addition, donnaMAMi also uses Onto4JaCa for querying in ontologies.

4.3 Onto4JaCa

Onto4JaCa[10] seeks to give intelligent agents, developed using the JaCaMo platform, the ability to use and manage the information contained in ontologies during their reasoning processes. It allows agents to access concepts, relationships and even semantic rules in OWL ontologies and reason using this information. It also allows agents to use the Pellet reasoner, a semantic reasoner, to make inferences about ontology information, understanding and communicating the explanations that came from the reasoner.

We have extended the approach presented by [FPH+17] to process the traces of computational steps (including the application of inference rules) used by semantics reasoners during queries to OWL ontologies. Then, using Onto4JaCa to translate those traces into an agent-oriented programming representation, agents can understand and manipulate that information, building explanations from external semantic reasoning. In addition, it is also possible for them to translate all the semantic rules contained in the ontology into argumentation schemes, in the format of defeasible rules, like those presented in Section 2.3.3, which are stored in their belief base and processed by agents using the framework presented in [PMB21, PB20].

Figure 4.3 presents an overview of the Onto4JaCa architecture. We implemented our ap-

[9] https://rasa.com/docs/

[10] `https://github.com/DeboraEngelmann/Onto4JaCa`

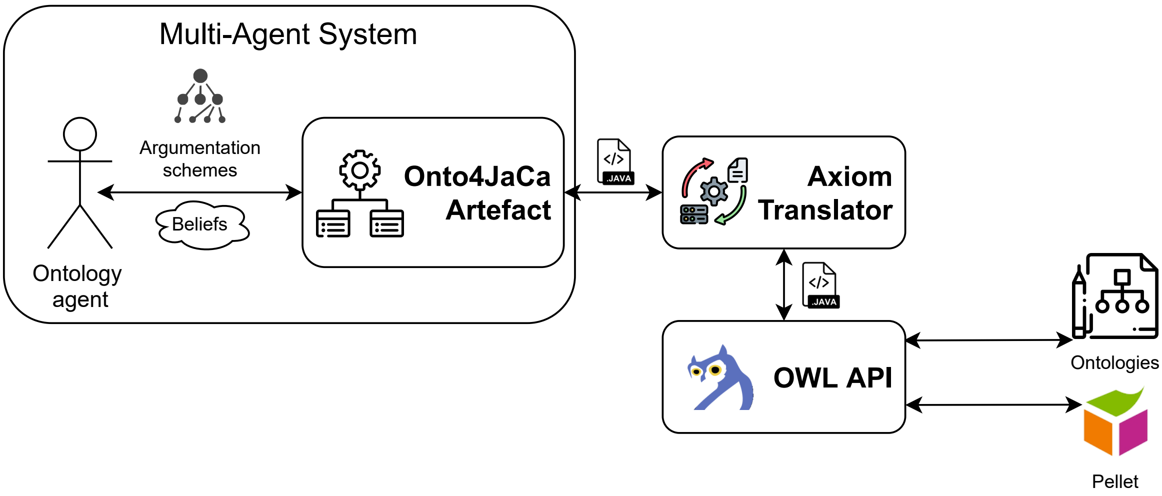

Figure 4.3: Onto4JaCa Architecture

proach using a CArtAgO artefact (named `Onto4JaCa Artefact`), which provides several internal operations to the agents. Among these internal operations, there are operations to extract concepts, relationships, data properties, individuals and SWRL rules from ontologies, as well as operations to add, delete or update this information in the ontology, and operations to seek inferences and explanations in semantic reasoners. When the agent triggers any of these internal operations in `Onto4JaCa Artefact`, we use the `OWL API`[11] (a Java API for creating, manipulating and serialising OWL Ontologies) as a basis for querying ontologies in conjunction with `Openllet` (an open-source OWL DL reasoner for Java based on Pellet) to extract the information from the ontology, as well as the explanations and inferences made by the semantic reasoner. `OWL API` returns this information in java object format. These objects are then translated into beliefs or defeasible rules by our `Axiom Translator` class and forwarded to `Onto4JaCa Artefact`, which makes them available to the agents.

When agents query semantic reasoners, they obtain the answer for the queries in the format of traces of computational steps (including concepts, classes and inference rules) used by the semantic reasoner to infer that particular query. Through the `Onto4JaCa Artefact`, 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 artefact 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,

[11] `https://github.com/owlcs/owlapi`

Table 4.1: Correspondence between Answers from Semantic Reasoners and an AOPL Representation [FPE+22].

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")`

the class *Axiom Translator* has the method *translateAxioms* that takes a set of objects of type *OWLAxiom* as a parameter, and, according to the type of that, converts each axiom to an AOPL representation. An example of this process is shown in Table 4.1. 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 showed in Section 2.3.3.

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 can build and communicate explanations represented in a computational representation for arguments. This is useful when the system requires agents to explain to other software agents. In order to provide explanations to human users, agents use natural language templates for argumentation schemes [PEB21, FPE+22] to translate those arguments to natural language arguments, then use those arguments to build and provide natural language explanations to human users.

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 SWRL rule below, which is available in the ontology used in a 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 reasoner, 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 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 to the representation required by the argumentation-based framework from [PMB21], 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. That means, after querying the ontology, the domain-specific rules used to answer that particular query are processed and translated to argumentation schemes and then stored into the agents' belief base.

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 for patient `patient2`, according to our running scenario, they are going to build an explanation using the computational representation for arguments introduced before.

```
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[12], agents query their belief base for the predicate they are interested in providing 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`. In this query, the argumentation-based framework 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)`.

Thus, for example, using the Jason plan presented in Listing 4, agents are able to explain a query to another software agent `Ag`.

```
+!buildExplanation(Query,Ag)
   :- argument(Query,Arg)
<- .send(Ag,explain,explanation(Query,Arg)).
```

Listing 4: Jason plan to explain a query

In this plan, an agent reacts to the triggering event `+!buildExplanation(Query,Ag)`, for example, when an agent creates the goal `!buildExplanation(ag,p)`, i.e., the agent has the goal of building an explanation of `p` for an agent `ag`. As a precondition to executing that plan, the agent must have an argument supporting that query, i.e., `argument(Query,Arg)`, so that the agent can proceed with the execution of the plan, sending the explanation to the target agent `Ag`, i.e., `.send(Ag,explain, explanation(Query,Arg))`.

Furthermore, an argument containing the answer for that particular query also is provided to the agent towards the interface developed. Consequently, agents become able to reason by themselves using the reasoning patterns extracted from the ontology (the inference rules

[12] We omitted the premises of argumentation schemes we did not present here. All argumentation schemes are available at`https://github.com/DeboraEngelmann/explaining-ontological-reasoning/blob/main/base_rules.md`

translated to argumentation schemes), obtained from previous queries, as well as, they can interpret, understand and communicate those answers as explanations.

When it is necessary to communicate with human users, agents can build natural language explanations, translating the computational representation of arguments to natural language arguments using natural language templates for argumentation schemes, as described in [FPE+22].

4.4 RV4JaCa

Communications between agents play a key role in the functioning of a multi-agent system since, in practice, agents rarely act alone; they usually inhabit an environment that contains other agents. Therefore, an extra layer of security that allows us to verify key aspects of this message exchange adds great value and great possibilities for improvement since certain aspects do not need to be considered when developing each agent. Using this type of formal verification at runtime allows us to standardise the interaction between agents through previously defined protocols that all agents must follow and, if they do not, react in a way that the execution is not negatively affected by the effects that were caused by this protocol deviation.

On the other hand, the verification done with RV is not limited to protocol validation. More specific properties of each application domain can also be verified once the monitor has access to the content of the exchanged messages. Even the execution of certain routines or functions can be executed according to the direction in which the agents' conversations go. For example, recording the results obtained during the agents' reasoning in a database without agents having the responsibility to carry out the registrations themselves. Or even sending an automatic email to a supervisor if any property identified by an agent and communicated to another is outside certain parameters. Therefore, depending on the MAS's domain, there is a range of possibilities in which RV can be used.

Based on that, we proposed RV4JaCa[13][EFP+22], a framework to integrate multi-agent systems and runtime verification. RV4JaCa allows performing runtime verification in multi-agent systems developed using the JaCaMo platform. In Figure 4.4, we present an overview of the approach. RV4JaCa is composed of the following: (i) a `Sniffer` class, developed in Java language, responsible for observing all communication between agents in the MAS; (ii) a CArtAgO Artefact named `RV4JaCa Artefact` responsible for analysing the messages perceived by the `Sniffer`, transforming them into a JSON object and sending it by a REST request to the `RML Monitor`. Note that RV4JaCa is not in any way limited to a specific kind of monitor; we used RML simply because it was the most suitable candidate for specifying the protocols of our interest. Nonetheless, a different monitor could be as easily integrated

[13] `https://github.com/DeboraEngelmann/RV4JaCa`

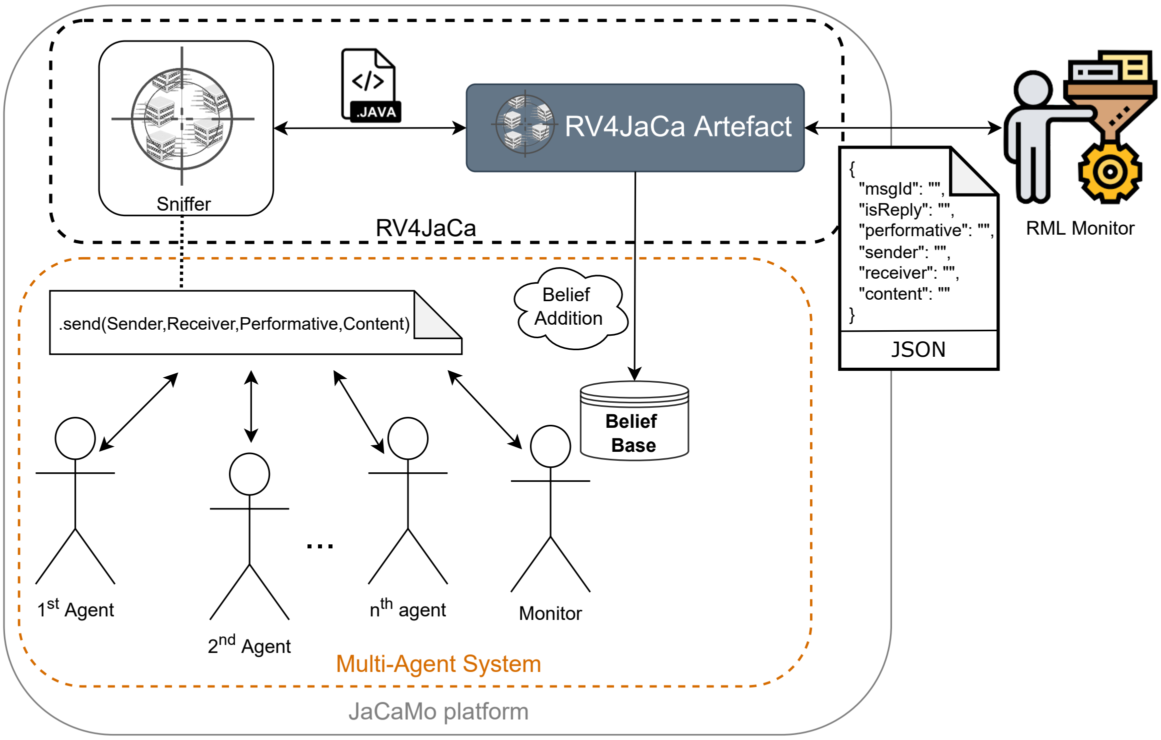

Figure 4.4: RV4JaCa Architecture

instead of the RML one. In addition, when the `RV4JaCa Artefact` receives the response of the request made to the `RML Monitor` saying that there was a violation, it can add a belief in the `Monitor` agent belief base; (iii) the `RML Monitor` responsible for analysing the events sent by `RV4JaCa Artefact` and verifying the satisfaction or violation of a formal property of interest; and (iv) a `Monitor` agent, which can be added to the system if it is necessary to interfere with agents' behaviour at runtime. In this case, if there is a violation, the `RV4JaCa Artefact` adds a belief to the `Monitor`'s belief base. When the agent perceives this addition, it can react by sending a message to the interested agents warning about the violation. This may trigger some consequent recovery mechanism, which usually is fully domain-dependent. On the other hand, the `Monitor` agent can also perform different activities depending on the system's needs.

In Section 5.4 we present a case study developed considering the bed allocation scenario to clarify how RV4JaCa works.

FINAL REMARKS

This chapter presented the overall architecture of the MAIDS framework we have developed. It is a modular framework to support the development of explainable multi-agent systems that can be used entirely, or it is also possible to use only the modules that fit the needs of each system under development. After presenting the overall architecture, each module is described in more detail, including some works that already use some of the modules we have developed. We have some publications related to this part of the work ([Eng20, EDK$^+$21b, EDK$^+$21a, EDP$^+$22, PEB21, FPE$^+$22, EFP$^+$22, OEK$^+$22]).

Chapter 5

Case Study

As a case study, we chose to implement a system using the MAIDS framework to assist in decision support in allocating hospital beds. We created some specific tools for this domain, as described below. We also conducted two evaluations with the professionals responsible for the allocation of beds at Hospital São Lucas da PUCRS in Brazil.

5.1 Bed Allocation Ontology

We use the inspiring approach [HJ02] to develop an ontology [ECG$^+$19] containing 95 classes, 85 object properties, 78 individuals, and 32 SWRL Rules (Semantic Web Rule Language). After the publication in [ECG$^+$19], we improve some details in our ontology. Now it contains 95 classes, 89 object properties, 81 individuals, and 40 SWRL Rules. We created the original version in Portuguese, aiming to use it in applications for Brazilian hospitals. We translated the terms here for consistency with the study report. Next, we explain its components (about the improved version) and present the rules we created based on the hospital bed management context. This ontology is available in a repository at GitHub[1]. Both versions of our ontology can be found in this repository: the published and improved versions.

[1] https://github.com/smart-pucrs/hospital-bed-allocation-ontology

Table 5.1: Bed allocation ontology object properties

Domain	Object Property	Range	Inverse of
Companion	accompanies	Patient	is-accompanied-by
Attendance	happens-in	Temporal_concept	
Nurse	allocates	Hospital_Bed	is-allocated-by
Health_Professional	analyses	Document	is-analysed-by
Patient	presents	Symptom	is-presented-by
Patient	presents-one	Disease	
Employee	attend	Patient	is-attended-by
Health_Professional	evaluates	Patient	is-evaluated-by
Patient	consumes	Medication	is-consumed-by
Patient	have-appointment	Doctor	
Patient	needs-assistance-like	Attendance	
Doctor	discharges	Patient	is-discharged-from
Patient	vacates-one	Hospital_Bed	is-vacated-by
Doctor	diagnoses-one	Patient	is-diagnosed-by
Attendance	is-associated-to	Risk_Classification	
State	is-assigned-to	Patient	
Hospitalisation	is-made-in-one	Hospital_Bed	
Patient	is-medicated-by	Nursing_Technician	
Patient	is-moved by	Health_Professional	
Patient	is-observed-by	Health_Professional	
Patient	is-classified-as	Risk_Classification	
Doctor	is-responsible-by	Patient	
Hospital_Bed	is-in	Bedroom	
Hospital_Bed	is-suitable-for	Patient	
Hospital_Bed	is-unsuitable-for	Patient	
Furniture	is-in-a	Local	
Bedroom	is-in-one	Hospitalisation_Unit	
Pharmacy	make-dispensation-of	Medication	
Cleaner	sanitises-one	Local	is-sanitised-by
Doctor	indicates-one	Treatment	
Patient	occupy-one	Hospital_Bed	is-occupied-for
Hospital_Bed	bed-may-have	Restriction	
Patient	may-have	Restriction	
Hospital_Bed	own-one	Situation	
Employee	fill-one	Document	is-filled-by
Doctor	prescribes	Medication	is-prescribed-by
Patient	has-one-attendance	Attendance	
Hospital_Bed	has-a	Bed	
Symptom	characterises-a	Disease	is-characterised-by

Our ontology is composed of 15 main classes, with 80 related sub-classes, containing 95 classes in total. Note that all the concepts described here refer to the scenario created by the researchers, and they can have different meanings in different contexts.

Attendance: The term *attendance* refers to the whole period the patient attended any activity in a hospital, either ambulatorial or hospitalisation. *Ambulatorial* is the assistance that occurs via prior scheduling or by emergency need. It has *Elective* (when it is scheduled) and *Emergency* sub-classes (when it occurs without scheduling because of an urgent need of the patient). *Hospitalisation* occurs when the patient needs to stay in hospital for more than one day, occupying a Hospital_Bed.

Risk_Classification: A risk category assigned to patients when they start being cared for is widely used in the emergency sector to prioritise patients in the worst health conditions. We describe it based on Manchester protocol, defined by Mackway-Jones, Marsden, and Windle [MJMW13]. Sub-classes range from immediate to non-urgent. A patient at risk of death is classified as *Immediate*. Patients with immediate risk of limb loss or loss of organ function are classified as *Very_urgent*. Patients with conditions that can worsen if not helped soon are *Urgent*. Patients with low risk of health damage are *Standard*, and those without any immediate risk of health damage are *Non_urgent*.

Temporal_concept: Concepts related to the timing of events. It includes the following sub-classes: Now, Year, Date, Day, Today, Hour, Time interval, Month, and Week.

Document: It refers to the documents generated during or after a patient's attendance. It includes diagnosis, report, prescription, and medical records. (1) *Diagnosis*: made by a doctor, it determines the disease's nature and causes, based on the patient history, symptoms, examination, etc. (2) *Report*: made by a specialist doctor, it usually contains the analysis of exams, such as radiology, laboratory, etc. (3) *Prescription*: made by a doctor, it includes drugs and treatments recommended to the patient. 4) *Medical_records*: it includes all the data related to the patient that can be accessed and stored by the hospital.

Disease: Biological alteration of a person's health state, manifested by a set of symptoms.

Speciality: Represents the medical specialisation or expertise that the doctor possesses or that the patient needs.

State: it represents patient conditions and has five sub-classes: Coma, In treatment, Stable, Severe, and Vegetative.

Situation: it represents hospital bed conditions and has five sub-classes: Blocked, Clean, Free, Occupied, and Dirty.

Local: places inside a hospital. Sub-classes are Corridor, Pharmacy, Hospital_Bed, Bedroom, Reception, Room and Hospitalisation_Unit. The Hospitalisation_Unit also has sub-classes named: Speciality_Unit, Nursery, Pediatrics, Intensive_Care_Unit, and Special_Care_Unit.

Medication: the drugs stored in the Pharmacy, which are meant to treat the patients.

Furniture: All the furniture that belongs to the hospital. For this study, we describe just two sub-classes: Bed and Stretcher.

Person: People who belong to the hospital ecosystem. Sub-classes are *Companion, Man, Woman, Patient,* and *Employee*. The employee has the following sub-classes: *Administration, Cleaner, Receptionist, Security_Guard, and Health_Professional*. The last one comprehends *Nurse, Nursing_Technician,* and *Doctor*. *Doctor* also have sub-classes named: *Generalist, Resident,* and *Specialist*. *Specialist* has the following sub-classes: *Cardiologist, Dermatologist, Neurologist, Oncologist, Pediatrician, Pneumologist, Radiologist, and Traumatologist*.

Restriction: rules to restrict bed allocation. (1) *Routing*: Origin of the patient, for example, if he came from the emergency or is an elective patient. (2) *Age*: person's age group, which can be adult, teenager, or child. (3) *Gender*: male or female. (4) *Isolation*: Refers to the cases where the patient cannot be in a room with other patients. (5) *Puerperal*: Women who just gave birth. (6) *Length_Of_Stay*: Predicted time of patient stay in the hospital, can be turn-fast or long-stay. (7) *Hospital_Care*: Hospital care the patient needs, can be surgical or clinical. (8) *Type_Of_Care*: The type of care the patient needs, can be minimal, semi-intensive or intensive.

Symptom: Signs to which the patient refers when talking about his illness (pain, fever, etc.).

Treatment: Set of instructions of procedures that the doctor recommends for the patient undergo.

We also created 89 relationships between the classes, and we present them in Table 5.1. We have not included all possible relationships between classes, only those we consider interesting for this domain, so we could see clearly how the classes relate to each other, to help us test the rules presented in Table 5.2.

We instantiated 81 individuals, so we could use the reasoner to test our rules. Individuals we created include Patient, Room, Hospital_Bed, Symptoms, and so on. To create them, we used names such as Patient1, 100, 100A, Headache, and so forth.

Our ontology aims to help in decision making about the beds where patients can be allocated according to the bed allocation constraints. Thus, we establish rules that propagate information about registered individuals' restrictions, and we can use these rules for ontological reasoning with Pellet reasoner. We are aware that many rules can be created to help in decision making related to bed allocation in hospitals. We present the ones we created for our ontology in Table 5.2.

Table 5.2: Rules of the bed allocation ontology

Rules	
1.	$\{Patient(?X), Man(?X) \rightarrow is-of-the-gender(?X, Male)\}$
2.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-isolation(?Y, ?I) \rightarrow bedroom-is-isolation(?Z, ?I)\}$
3.	$\{Patient(?X), has-one-attendance(?X, ?A) \rightarrow is-routing(?X, ?A)\}$
4.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-the-attendance(?Z, ?A)$ $\rightarrow bed-is-the-attendance(?Y, ?A)\}$
5.	$\{Patient(?X), is-routing(?X, ?E), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-routing(?Y, ?E)\}$
6.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-stay(?Y, ?P) \rightarrow bedroom-is-stay(?Z, ?P)\}$
7.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-puerperal(?Y, ?Q) \rightarrow bedroom-is-puerperal(?Z, ?Q)\}$
8.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-speciality(?Z, ?S) \rightarrow bed-is-speciality(?Y, ?S)\}$
9.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-speciality(?Y, ?S) \rightarrow bedroom-is-speciality(?Z, ?S)\}$
10.	$\{Patient(?X), Woman(?X) \rightarrow is-of-the-gender(?X, Female)\}$
11.	$\{Patient(?X), is-care(?X, ?C), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-care(?Y, ?C)\}$
12.	$\{Patient(?X), is-puerperal(?X, ?Q), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-puerperal(?Y, ?Q)\}$
13.	$\{Patient(?X), is-the-attendance(?X, ?A), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-the-attendance(?Y, ?A)\}$
14.	$\{Patient(?X), is-stay(?X, ?P), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-stay(?Y, ?P)\}$
15.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-routing(?Y, ?E) \rightarrow bedroom-is-routing(?Z, ?E)\}$
16.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-of-the-age-group(?Y, ?G)$ $\rightarrow bedroom-is-of-the-age-group(?Z, ?G)\}$
17.	$\{Patient(?X), is-classified-as(?X, Urgent) \rightarrow needs-assistance-like(?X, Hospitalisation)\}$
18.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-of-the-age-group(?Z, ?G)$ $\rightarrow bed-is-of-the-age-group(?Y, ?G)\}$
19.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-of-the-gender(?Z, ?H) \rightarrow bed-is-of-the-gender(?Y, ?H)\}$
20.	$\{Patient(?X), is-isolation(?X, ?I), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-isolation(?Y, ?I)\}$
21.	$\{Patient(?X), is-speciality(?X, ?S), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-speciality(?Y, ?S)\}$
22.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-routing(?Z, ?E) \rightarrow bed-is-routing(?Y, ?E)\}$
23.	$\{Patient(?X), is-of-the-gender(?X, ?H), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-of-the-gender(?Y, ?H)\}$
24.	$\{Patient(?X), is-of-the-age-group(?X, ?G), Hospital\_Bed(?Y), occupy-one(?X, ?Y) \rightarrow bed-is-of-the-age-group(?Y, ?G)\}$
25.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-the-attendance(?Y, ?A) \rightarrow bedroom-is-the-attendance(?Z, ?A)\}$
26.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-care(?Z, ?C) \rightarrow bed-is-care(?Y, ?C)\}$
27.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-isolation(?Z, ?I) \rightarrow bed-is-isolation(?Y, ?I)\}$
28.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-puerperal(?Z, ?Q) \rightarrow bed-is-puerperal(?Y, ?Q)\}$
29.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-of-the-gender(?Y, ?H) \rightarrow bedroom-is-of-the-gender(?Z, ?H)\}$
30.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bedroom-is-stay(?Z, ?P) \rightarrow bed-is-stay(?Y, ?P)\}$
31.	$\{Hospital\_Bed(?Y), Bedroom(?Z), is-in(?Y, ?Z), bed-is-care(?Y, ?C) \rightarrow bedroom-is-care(?Z, ?C)\}$
32.	$\{Patient(?X), is-years-old(?X, ?Y), lessThan(?Y, 12) \rightarrow is-of-the-age-group(?X, Child)\}$
33.	$\{Patient(?X), Hospital\_Bed(?Y), is-of-the-gender(?X, ?Gp), bed-is-of-the-gender(?Y, ?Gb),$ $DifferentFrom(?Gp, ?Gb) \rightarrow is-unsuitable-for(?Y, ?X)\}$
34.	$\{Patient(?X), is-discharged-from(?X, ?Y), occupy-one(?X, ?Z) \rightarrow vacates-one(?X, ?Z), is-vacated-by(?Z, ?X)\}$
35.	$\{Patient(?X), Hospital\_Bed(?Y), is-speciality(?X, ?Sp), bed-is-speciality(?Y, ?Sb),$ $DifferentFrom(?Sp, ?Sb) \rightarrow is-unsuitable-for(?Y, ?X)\}$
36.	$\{Patient(?X), is-years-old(?X, ?A), greaterThan(?A, 17) \rightarrow is-of-the-age-group(?X, Adult)\}$
37.	$\{Patient(?X), Hospital\_Bed(?Y), is-of-the-age-group(?X, ?Ap), bed-is-of-the-age-group(?Y, ?Ab),$ $DifferentFrom(?Ap, ?Ab) \rightarrow is-unsuitable-for(?Y, ?X)\}$
38.	$\{Patient(?X), Hospital\_Bed(?Y), is-of-the-gender(?X, ?G), bed-is-of-the-gender(?Y, ?G),$ $is-of-the-age-group(?X, ?A), bed-is-of-the-age-group(?Y, ?A), is-speciality(?X, ?S), bed-is-speciality(?Y, ?S),$ $is-care(?X, ?C), bed-is-care(?Y, ?C) \rightarrow is-suitable-for(?Y, ?X)\}$
39.	$\{Patient(?X), is-years-old(?X, ?A), greaterThan(?A, 11), lessThan(?A, 18) \rightarrow is-of-the-age-group(?X, Teenager)\}$
40.	$\{Patient(?X), Hospital\_Bed(?Y), is-care(?X, ?Cp), bed-is-care(?Y, ?Cl), DifferentFrom(?Cp, ?Cl)$ $\rightarrow is-unsuitable-for(?Y, ?X)\}$

5.2 Bed Allocation Optimisation

In our previous work [Eng19], one of the improvements suggested by the professionals responsible for bed allocation who evaluated our application was that our agent could help optimise bed allocation, suggesting beds for patients to be allocated. In order to meet this request, we created an optimiser for bed allocation[2]. For this purpose, we chose to use the simplex method since it is a popular method of solving linear programming systems.

The bed allocation optimisation program takes the database's restrictions and converts them into three types of linear restrictions, equality, relative equality, and negation.

```
(2*Q112[p])+(abs(gender[p]-2)/2) <= 2;
```

The equality constraint requires that every patient allocated to the bedroom has a specific characteristic. In this case, we want the patient to have gender 2. Q112[p] is a Boolean value equal to 1 if patient p is in the bedroom; gender[p] is an integer value referring to the patient's gender. The left side of the sum results in 2 if the patient has been allocated to that bedroom; otherwise, it results in 0. The right side results in 0 if the patient's gender is 2; otherwise, it results in some number greater than zero. The result of the sum must be less than or equal to two, being the only case in which this does not happen when the patient is allocated to the bedroom, but the gender is not two.

```
(Q112[p1]+Q112[p2])+(abs(gender[p1]-gender[p2])/2) <= 2;
```

Relative equality requires that if two patients are allocated to the same bedroom, they must have the same characteristic; in this case, the same gender. The sum's left side has a maximum value of 2 when the two patients are in room 112. The right side results in a value greater than zero when the two patients' genders differ. The restriction is not obeyed only when the two patients are in the same bedroom, but the genders are different.

```
(Q112[p]) - abs(gender[p]-2) <= 0;
```

Negation requires that every patient allocated to the bedroom does not have a specific characteristic. If the patient is in the bedroom, but the gender is 2, this will result in 1-0 = 1, which does not comply with the restriction. If the patient is not allocated to the room or the gender is not 2, the result is less than or equal to zero, following the bed's rules.

[2] This application was developed jointly with the student of scientific initiation Lucca Dornelles Cezar in the project "Seleção e integração de técnicas de Teoria da Argumentação para o contexto hospitalar", Edital FAPERGS 03/2020 – PROBIC/PROBITI.

For this development[3], we are using GLPSol[4] (Gnu Linear Programming Solver). It is a free, open-source program for solving linear programming problems. One of the implemented algorithms is the simplex method. GLPSol allows the user to set certain limits, such as a time limit. When the limit is reached, the process returns the best result found. This allows the program to generate some possibly useful allocation suggestions without arriving at the optimal result, which can be an extremely time-consuming process in some instances.

We have integrated the optimiser with our multi-agent system. Figure 5.1 presents a result of an allocation made using the optimiser[5].

5.3 Plan Validator

When real-world problems can be modelled in a planning language, it is possible to use a plan validator to tell the human operator whether the plan is feasible or not [HLF04]. Behnke *et al.* [BHB17] define plan verification as "the task of determining whether a plan is a solution to a given planning problem". A plan validator can be used in a wide range of applications. The application that interests us is the validation of bed allocation plans prepared by the user.

In our previous work [Eng19], we use a plan validation tool for PDDL (Planning Domain Definition Language) called VAL [HLF04] to check if any bed allocation rule has been broken in the user allocation. However, in this work, to achieve better performance, we developed a new plan validator using Java[6]. Like VAL, our plan validator also receives three PDDL files as input: a file containing the domain, a file containing the problem and a file containing the plan to be validated.

The domain file establishes some basic rules, such as the types of objects and possible actions. The actions are generally divided into three parts, the parameters, the preconditions, and the effects. In this case, the action of allocating a patient to a bed requires that the patient is not allocated to another bed and that the bed is empty. The effects generated by applying this action are that the patient is now allocated, the room is occupied, and the patient is allocated in that room. Problem files use domain rules to complete a problem, making all objects (e.g. patients and beds) and objectives explicit (e.g., all patients must be allocated).

For validators, a plan file is also necessary, which is simply a set of actions to be applied

[3] `https://github.com/smart-pucrs/explainable_agents/tree/glpsol`

[4] http://winglpk.sourceforge.net/

[5] All patient data in our tests are fictitious

[6] This application was developed jointly with the student of scientific initiation Lucca Dornelles Cezar in the project "Seleção e integração de técnicas de Teoria da Argumentação para o contexto hospitalar", Edital FAPERGS 03/2020 – PROBIC/PROBITI.

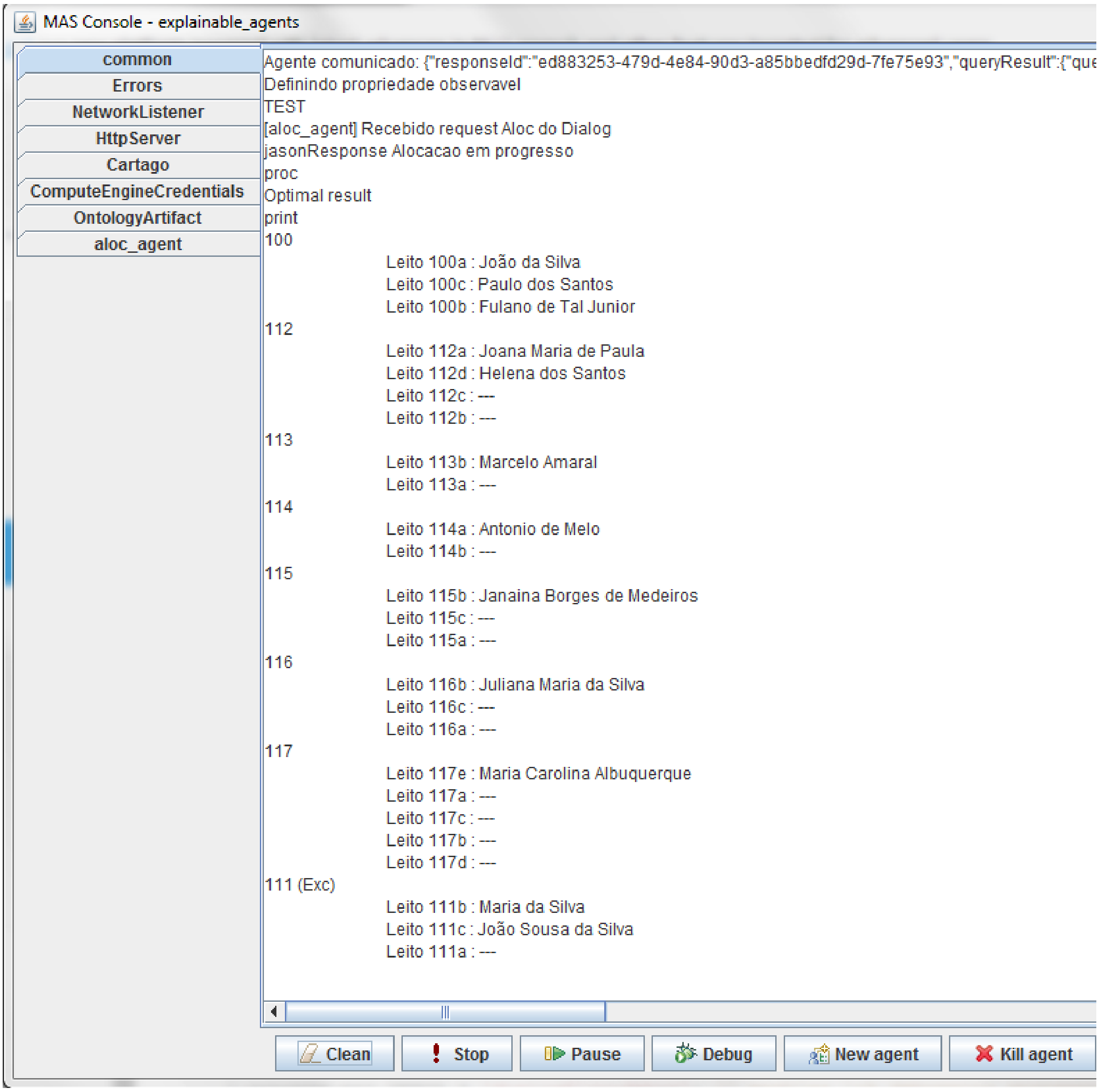

Figure 5.1: Optimiser output

sequentially, which lead to the objective of the problem. Given these files, the validation process is straightforward, check if each action is applicable and then apply its effects. An action is only applicable if the types of parameters are correct (e.g., patient1 bed3) and if the preconditions have been met. If any action is not applicable, the plan is considered flawed. If all actions are applicable, but the problem's objective is not satisfied, then the plan is also considered to be flawed. The plan only satisfies the problem if all actions are applicable and the objective is satisfied after applying the last action. A characteristic of the hospital domain is that the actions are not ordered. It is possible to generate several changes in the order of execution of the actions since they are actions independent of each other, which, in general, would not be possible in other domains.

In our plan validator, the user can print the plan and the validation result in the terminal through the *planTest* function as shown in Figure 5.2. The user still has the possibility, through the *valOut("filename")* function, to obtain a LaTeX file that generates a PDF with the validation report according to Appendix A. These results presented in Figure 5.2 and Appendix A are based on domain, problem and plan files adapted from our previous work [Eng19].

As the program was designed thinking about a specific type of problem, we have considered certain restrictions to facilitate the implementation of this version. In this version, we have some limitations that do not negatively affect the results of the tasks for which we are using the validator but that need to be resolved to make our validator available to the research community to use it. Although the PDDL language is modular, specific options are practically universally accepted. However, not all have been implemented in our validator. In total, the program supports three types of requirements:

- “STRIPS”, which is necessary to use the program;
- “typing”, which allows the use of types and sub-types, but sub-types have not been implemented; and
- “equality”, which allows the use of equality comparisons.

Another actual limitation is that the planner was made to be used in automatically generated problems. The plan validator considers that the PDDL files are semantically and syntactically correct. Because of that, we have not implemented any pre-processing to look for errors. The current version of our plan validator is available on GitHub[7].

[7] https://github.com/smart-pucrs/PDDL-plan-validator

```
DOMAIN hospital
requirements
        :typing
        :equality
        :strips
types
        genero
        tipoDeEncaminhamento
        leito
        paciente
Actions
        aloc-Agudo
        Params:  ?paciente [paciente] ?leito [leito]
        NegPrecd:  [ocupado ?leito][alocado ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

        aloc-Eletivo
        Params:  ?paciente [paciente] ?leito [leito]
        NegPrecd:  [ocupado ?leito][alocado ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

        aloc-Neurologia
        Params:  ?paciente [paciente] ?leito [leito] ?genero-paciente [genero] ?
genero-leito [genero]
        PosPrecd:  [= ?genero-paciente ?genero-leito]
        NegPrecd:  [ocupado ?leito][alocado ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

        aloc-Medicina-Interna
        Params:  ?paciente [paciente] ?leito [leito] ?genero-paciente [genero] ?
genero-leito [genero]
        PosPrecd:  [= ?genero-paciente ?genero-leito]
        NegPrecd:  [tipoDeEncaminhamento-_NONE ?paciente][ocupado ?leito][alocad
o ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

        aloc-Cardiologia
        Params:  ?paciente [paciente] ?leito [leito] ?genero-paciente [genero] ?
genero-leito [genero]
        PosPrecd:  [= ?genero-paciente ?genero-leito]
        NegPrecd:  [ocupado ?leito][alocado ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

        aloc-Oncologia
        Params:  ?paciente [paciente] ?leito [leito] ?genero-paciente [genero] ?
genero-leito [genero]
        PosPrecd:  [= ?genero-paciente ?genero-leito]
        NegPrecd:  [ocupado ?leito][alocado ?paciente]
        Positive Effects:  [alocado ?paciente][ocupado ?leito][in ?paciente ?lei
to]

predicates
        genero-Feminino [?paciente - 3]
        genero-Masculino [?paciente - 3]
        tipoDeEncaminhamento-_NONE [?paciente - 3]
        tipoDeEncaminhamento-Agudo [?paciente - 3]
        tipoDeEncaminhamento-Eletivo [?paciente - 3]
        alocado [?paciente - 3]
        ocupado [?leito - 2]
        in [?paciente - 3] [?leito - 2]
__________________________
Goal achieved
```

Figure 5.2: Validation report

5.4 Runtime Verification in Bed Allocation

We developed a case study considering the bed allocation scenario to clarify how the RV4JaCa works. This case study includes the following agents:

assistant: the internal representation in MAS for a chatbot that assists hospital staff in carrying out bed allocation in a hospital;

operator: the internal representation in MAS for the hospital staff member who operates the system for allocating beds;

database: an agent responsible for querying and updating the bed allocation system database.

validator: an agent responsible for validating bed allocation plans using a PDDL plan validator.

optimiser: an agent responsible for making suggestions for optimised allocations using the GLPSol solver of GLPK.

monitor: an agent responsible for warning the agents involved in the exchange of messages when there has been a violation in a property.

In this case study, RV4JaCa is responsible for collecting information about all messages exchanged between agents and sending them through a REST request to the RML monitor. After processing the received message, the monitor returns a result that identifies whether the message sent from one agent to the other violates any of the properties being checked by it. If a property is violated, RV4JaCa adds this information to the `Monitor` agent's belief base, which in turn warns the agents involved in the exchange of messages that there has been a violation. This makes it possible for our agents to take action to recover from the failure that the breach caused.

The ability to monitor the messages exchanged between the agents in this case study is used for two different purposes. The first one is, according to the performative and content of each move, to verify whether the agents are following the predefined communication protocol. This aspect may be crucial for safety-critical and privacy-preserving aspects. For instance, in a healthcare domain, such as the bed allocation one, the agents might be expected to follow specific medical guidelines for communicating personal information (even amongst themselves). Moreover, when in the presence of multiple agents, each with its own goals, it is common to specify the ideal expected outcome at a more abstract level, where it is more natural to reason upon.

The second one is to check if the human changed the topic of the conversation without the proper conclusion of the previous topic. When we add humans to the agent-to-agent communication loop, the developer has limited control over the human interactions in the dialogue. For example, in some cases, when the MAS is performing some specific task, it is important to be able to finish it before starting a new one. But when completing this task depends on some human interaction, we have no guarantee that the human will complete the necessary exchange. Naturally, it would be possible to do this check within each agent. However, when we have agents specialised in specific tasks, it is preferable to assume the messages handled by the agents are all related to the current topic; so to focus on the actual agents' implementation.

In the following, we report two example properties, written in RML, that have been checked for the bed allocation case study using RV4JaCa.

5.4.1 First RML Property for the Bed Allocation Domain

The first property, which is presented in Listing 5.1, concerns checking the user does not change the topic before completing the one currently processed by the agents. In particular, the reported property cares about the `'getValidationResult'` topic. Such topic relates to the user asking the assistant agent to validate a suggested bed allocation (the corresponding event type is expressed in lines 1-5). When the assistant receives this message, the protocol goes on, causing a sequence of messages exchanged amongst the assistant, the optimiser and the validator agents. Note that this part is not reported because it is not of interest for checking the property. After this step, the assistant agent sends back an answer to the user (the event type is in lines 6-11). Suppose this answer is not empty (*i.e.*, the event contains `arg1`, and `arg2` fields). In that case, the user is expected to conclude the communication with a certain content (listed in lines 12-17). For instance, the user could reply with an additional message containing `'allocValPatients'`, meaning that the user wants to allocate all patients according to the plan, even if there are failures because the reported failures correspond to rules he is aware of breaking. Naturally, the user might decide not to complete any of the plan allocations that have been validated, as some have flaws. In that scenario, the message would have content `'dontAllocValPatients'`. Another possibility is the message has the content `'allocValidValPatients'`, meaning that the user wants to allocate only those patients where the allocation does not break any rules. And the last one is when the user asks for an optimised bed allocation suggestion. In this case, the message would be content `'getOptimisedAllocation'`.

Once the events corresponding to previously mentioned messages have been specified (lines 1-26), the actual property can be expressed following the RML syntax (lines 27-30). In more detail, in line 27, we may find the definition of the main term denoting the property to check (which in RML is always called `Main`). In this scenario, the principal term corresponds to a sequence of subterms, named `Question`. Such term is defined in line 28 and starts with a question (as defined in lines 1-5). This means that to comply with the protocol, the first event has to be a message containing a `'question'` of the `'operator'` for the `'assistant'` (in this case regarding the validation of a bed allocation). After that, the property goes on with the `Answer` term (line 29). Inside it, we find a disjunction between two possible alternatives in the protocol. On the left, we may observe a `answer_with_constraint` event, which means, according to lines 6-11, that the `'assistant'` replied to the `'operator'` with a suggested result that the latter has to decide upon. On the right, we may observe an event corresponding to any other answer, which in this specific case denotes the case where the result is empty, meaning that no result is available to be suggested to `'operator'`. In the latter case (the right branch), the property ends this cycle, since the communication between the two agents is concluded, and new messages concerning new topics can be exchanged in the future. Instead, in the former case (the left branch), the current cycle is not ended, because the `'assistant'` is still waiting for an answer from the `'operator'` regarding the suggested result. This last aspect is handled in the term in line 30, where not any question is admissible from the `'operator'`, but only one amongst the ones listed in lines 12-17. Upon the reception of the event matching one such listed event type, the property cycle ends, and as it happened for the right branch, the protocol can move on.

Now, before presenting another property of interest that we have analysed through RV4JaCa, it is important to detail how a property can be violated. As we mentioned before, we presented the events that are accepted in certain points of the property and why. An RML property is violated whenever, given the current term denoting the current state of the property, and a new event, the property does not accept the such event. For instance, in the property presented in Listing 5.1, an event which is different from an answer, after having observed a question, is not acceptable. This can be seen in line 29, where after consuming an event denoting a question, the only possible following events can be an answer requiring additional info (left branch), or a general answer (right branch). Thus, if the observed event is neither of the two, the term is stuck and cannot move on. In RML, this translates into a violation of the property, which is then reported back to the monitor agent, that in turn, will trigger all mechanisms for the agents involved to react properly.

5.4.2 Second RML Property for the Bed Allocation Domain

The second RML property we tested in the bed allocation domain is reported in Listing 5.2. Differently from the property reported in Listing 5.1, here, we do not check the consistency amongst topics. Instead, we care about checking that an agent always replies to a question, before posing a new one. As before, the first part of Listing 5.2 concerns the definition of which events are of interest for the property (lines 1-10). In this specific case, we have questions (lines 1-5), and answers (lines 6-10). Note that, differently from the previous RML property, here we exploit parameters inside the specification. Indeed, the event types reported in lines 1-10 are all parametric w.r.t. the agents involved in the communication. This means that such event types do not focus on specific messages exchanged between predefined agents, as it was before, but are free (through RML parameters, we have late

```
1  question matches {
2      performative:'question',
3      sender:'operator', receiver:'assistant',
4      content:{name:'getValidationResult'}
5  };
6  answer_with_constraint matches
7  {
8      performative:'assert',
9      sender:'assistant', receiver:'operator',
10     content:{name:'answer', name:'result', arg1:_, arg2:_}
11 };
12 constrained_question matches
13     {performative:'question', sender:'operator', receiver:'assistant', content:{name:'allocValPatients'}}
        |
14     {performative:'question', sender:'operator', receiver:'assistant',
        content:{name:'getOptimisedAllocation'}} |
15     {performative:'question', sender:'operator', receiver:'assistant',
        content:{name:'dontAllocValPatients'}} |
16     {performative:'question', sender:'operator', receiver:'assistant',
        content:{name:'allocValidValPatients'}};
17 a_question matches
18 {
19     performative:'question',
20     sender:'operator', receiver:'assistant'
21 };
22 an_answer matches
23 {
24     performative:'assert',
25     sender:'assistant', receiver:'operator'
26 };
27 Main = Question*;
28 Question = (question Answer);
29 Answer = (answer_with_constraint ConstrainedQuestion) \/ (an_answer);
30 ConstrainedQuestion = constrained_question Answer;
```

Listing 5.1: The RML specification for checking that no change of topic is observed after the user has requested a validation result.

binding on the agents involved in the interaction). This makes the definition of the RML property in line 11 highly parametric, without updating the property for each new agent added to the system. The property is defined in line 11, through the standard `Main` term in RML. Since the property is parametric, it starts with the `let` operator, which defines the variables used in the term. In this case, the variables used are `ag1`, and `ag2` (naturally, any other name would have sufficed). After that, the property goes on, expecting a question, followed by a corresponding answer. Here, note that in the first event (*i.e.*, the question), the variables are bound to the agents involved in the communication, while in the second event (*i.e.*, the answer), such variables are ground to the previously initialised values. In this way, a question is free to be sent by any possible agent `ag1`, to any possible agent `ag2` in the system (where `ag1` and `ag2` are bound to the observed agents involved in the communication); instead, an answer is constrained to be sent by agent `ag2` to agent `ag1` (with both variables already bound to the respective values through the previously observed question).

```
question(ag1, ag2) matches
{
    performative:'question',
    sender:ag1, receiver:ag2
};
answer(ag1, ag2) matches
{
    performative:'assert',
    sender:ag1, receiver:ag2
};
Main = {let ag1, ag2; question(ag1, ag2) answer(ag2, ag1)}*;
```

Listing 5.2: The RML specification for checking that an agent always replies before starting messaging about something else.

As before, also with this property, we can ponder on which events can cause a violation. In particular, the property expressed in Listing 5.2 is violated when after a question between two agents (ag1⟶ag2), the following event is not the corresponding answer (ag2⟶ag1), but another message (for instance another question).

5.5 Evaluation

Hospital São Lucas da PUCRS in Brazil has kindly agreed to support us in evaluating our system. We divided the evaluation of our system into two parts. The first one focused on system functionality. The aim was to verify if the functionalities that the decision support system developed using the MAIDS framework were adequate to the needs of the professionals responsible for allocating beds in the hospital. The second one was intended to verify the expressiveness of our framework when using real data in a real-world domain. In both evaluations, this case study included the following agents:

assistant (a): the internal representation in MAS for a chatbot that assists hospital staff in carrying out bed allocation in a hospital;

ontology (l): an agent with access to ontologies responsible for semantic reasoning using argumentation schemes as defeasible rules generated automatically from the semantic rules contained in the ontology.

operator (o): the internal representation in MAS for the hospital staff member who operates the system for allocating beds;

nurse (n): the internal representation in MAS for a nurse who in that hospital serves as domain expert for bed allocation and whom the operator needs to consult in case of doubt;

database (d): an agent with access to the hospital's general information system for checking details of past and current patients, bed allocations, etc.

validator (v): an agent responsible for validating bed allocation plans using the PDDL plan validator presented in Section 5.3.

optimiser (p): an agent responsible for making suggestions for optimised allocations using the optimiser presented in Section 5.2.

Furthermore, in both evaluations, were used the modules Dial4JaCa (4.2) for communication between the multi-agent system and Dialogflow; Onto4JaCa (4.3) to access ontology information and rules; and RV4JaCa (4.4) to observe all conversations between agents and generate logs of these conversations so that we can later test using the RV4JaCa module in its entirety. Below, both evaluations performed with hospital staff are described in more detail.

5.5.1 Evaluation of System Functionalities for Hospital Bed Allocation

In this evaluation, we seek to assess whether changes will be necessary to adapt the dialogue system instance created from the MAIDS framework to be used with real data from that hospital and whether the functionalities already developed were in accordance with their needs. We fed the web interface with fictitious data about beds and patients. Based on the data in the system, we asked that professionals use the simulator to check out the fictitious hospital situation and ask the chatbot to validate the bed allocation they created, give suggestions, evaluate the availability of a bed related to a specific patient, and explain the statements put forward. Afterwards, we asked the professionals to evaluate the chatbot's answers. Also, we performed an online questionnaire to collect the opinion of these professionals about the use of the system. All professionals signed a consent form for participation.

5.5.1.1 Evaluation Procedure

We created fictitious data based on our previous research that contained patients with their characteristics, beds with their status, bed requests, and exceptions. During testing, we used 40 beds, 35 inpatients, and 6 bed requests. Of the 40 beds, there were vacant and occupied statuses. We create an ontology with the following settings: Classes (95), Object Properties (51), Data Properties (3), Individuals (8), Datatypes (2), and SWRL Rules (5).

Two hospital staff responded to our questionnaire. The first one has been a bed management administrator for nine years. Moreover, the second one has been the medical coordinator in this hospital for one year and is one of the doctors who assisted in the construction of a manual for the implantation and implementation of the internal regulation committee (including bed-allocation rules) for general and specialised hospitals [dS17] used by many hospitals in the country. Table 5.3 presents the specialists' demography.

Table 5.3: Specialists demography

Id	**Gender**	**Job occupation**	**How long in this job?**
S1	M	Bed management administrator	9 year
S2	M	Medical coordinator	1 year

The evaluation was performed online and asynchronously with the two specialists. We sent them a video showing the system's functionalities and explaining the intention of this evaluation. They could access the system online and perform the tests by themselves. After the test, the experts answered an online questionnaire. We created a questionnaire with eight closed questions following the 5-point Likert scale to evaluate the functionalities (Table 5.4) and an open question to suggestions. Also, we generate a system log with all requests made as presented in Appendix B.

Table 5.4: Questions to evaluate the functionalities

Id	**Question**
Q1	Are the bed allocation rules considered in the development of the software consistent with those practised in the hospital?
Q2	Are the answers given by the chatbot easily understandable?
Q3	Are the chatbot's allocation suggestions adequate according to the patients' needs?
Q4	Do the suggestions for optimised allocations contemplate the allocation rules, and are they considered adequate to be used?
Q5	Is the chatbot able to validate a bed allocation and still explain which allocation rules are being broken when the allocation is not valid?
Q6	When asked if a bed is suitable for a patient, is the chatbot able to respond and still explain in an easily understandable way how it reached that?
Q7	Is the option to request an optimised allocation by moving a maximum number of patients from the beds where they are currently allocated useful daily?
Q8	Is the option of registering exceptions so that the system considers them when making suggestions useful daily?

5.5.1.2 Results

As we can see in Figure 5.3, the questionnaire results indicated that some rules would need to be added or changed. The evaluators asked us to add rules related to patients with infection, information about health insurance plans or health plans, and information sent

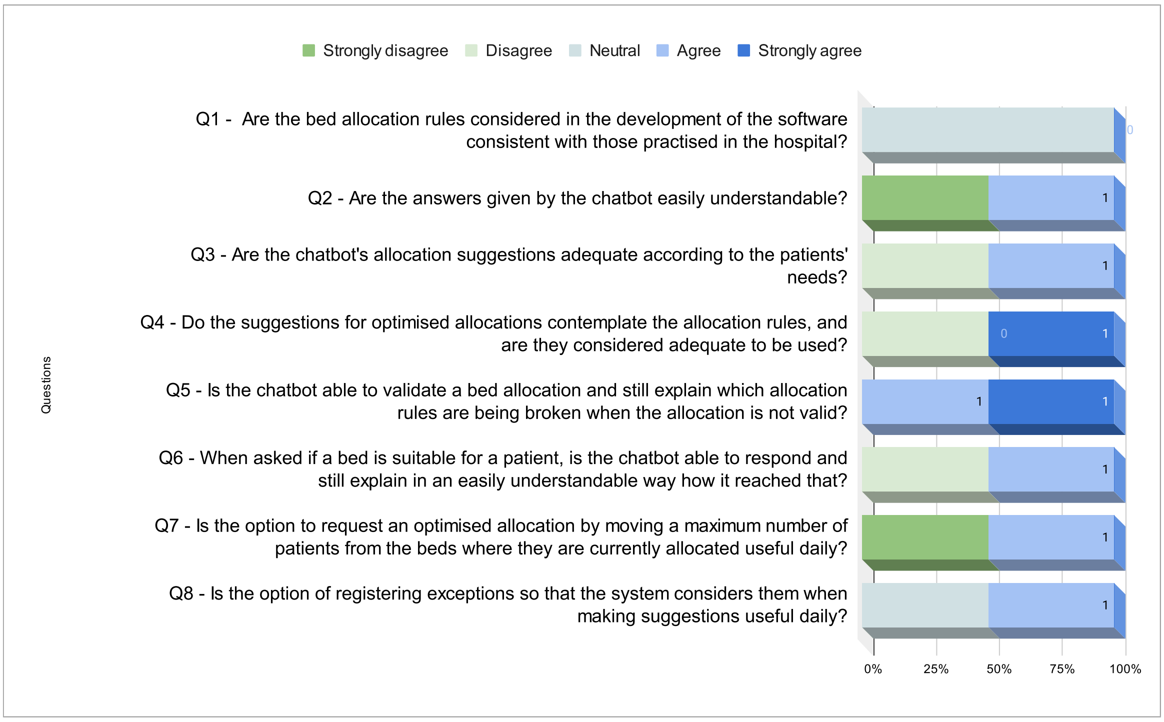

Figure 5.3: Evaluation result

by the bed requesting unit. Due to inconsistencies between the rules used by the agents and those used in the hospital, the interaction with the chatbot was also compromised since the explanations it gave sometimes did not match the reasons used in real life. Because of that, one evaluator strongly disagrees that the answers given by the chatbot are easily understandable. On the other hand, both professionals agree that the chatbot is able to validate the bed allocation and explain which allocation rules are being broken when that is the case. In addition, one of the professionals agrees that when asked if a bed is suitable for a patient, the chatbot can answer and also explain how it reached that conclusion in an easily understandable way, while the other one disagrees.

As a consequence of this evaluation, the local (university) hospital managers have asked us to help deploy our multi-agent system to be used in their daily bed management activities as soon as we can interface it with the information systems currently used in the hospital.

5.5.2 Evaluation of Expressiveness of the Framework

Some desiderata for task-oriented dialogue systems have been recently formulated [Coh19]. We summarise that desiderata below and give in parenthesis the lines of an example dialogue using our framework where each of the features of the desiderata is demonstrated. The example also illustrates the ontological subdialogues supported by our framework.

1. The system should allow the explicit representation of the user's desires that are implicit in requests such as in (1).
2. The system should be able to represent the meaning of users' utterances in logical forms, including constraints having two superlative expressions, one embedded within the other as exemplified in (8 and 27).
3. In the case of multiparty dialogues, it should keep track of the mental attitudes of all the involved participants as in (9 and 19).
4. It is important to reason about plans and intentions, as it allows the system to be helpful by reasoning about what the user is trying to do, as in (18–25).
5. It should reason about the meaning of mental attitudes as in (1 and 22).
6. It should also represent other agents' beliefs without having precise information about what those beliefs are (9).

The idea behind such desiderata is to have a system that is fully explainable because everything it says has an explicitly represented plan being referred to by the system.

We now reproduce an example dialogue involving both humans and agents, including a version of the dialogue system that supports natural language interaction through the use of Dialogflow and has been developed and evaluated with the support of medical staff from Hospital São Lucas da PUCRS in Brazil.

These excerpts exemplify the type of dialogues that can take place in systems developed with the approach put forward in this thesis. They demonstrate the ontological discussions (in lines 17–17p) and the desired features discussed above.

The dialogue starts with the operator trying to allocate a bed to a particular patient and proceeds as follows. We show each (numbered) dialogue game move, but *before* it we provide an English equivalent for readability. We enclose in curly brackets the changing beliefs of some of the agents which underlie the dialogue move. Note that our approach only allows for atomic formulæ in argument conclusions, but it allows for constraints on a particular conclusion to be specified using Jason annotations, so if a dialogue move contains a formula `p(X)[q(X)]`, it means that in Jason we will find an instantiation for `X` such that `p(X) & q(X)` holds.

operator to assistant: check if any female surgical bed is free;
`1. question(o,a,free(B)[female(B), surgical(B)])`

{assistant: `des(o,allocate(P,B)[female(B),surgical(B)]), bel(d,free(B))`}
(desiderata (1 and 5))
assistant to database: check if any female surgical bed is free;
`2. question(a,d,free(B)[female(B),surgical(B)])`
database to assistant: bed 203b is free and satisfies those constraints;
`3. assert(d,a,free(203b)[female(203b),surgical(203b)])`
assistant to operator: bed 203b has the characteristics you requested;
`4. assert(a,o,answer(free(203b)[female(203b),surgical(203b)])`
operator to nurse: I'm allocating Patient8 to 203b;
`5. assert(o,n,allocate(patient8,203b))`
{assistant: `des(o,allocate(patient8,B)[female(B),surgical(B)])`}
nurse to operator: we cannot allocate patient8 to 203b;
`6. assert(n,o,~allocate(patient8,203b))`
operator to nurse: why?
`7. challenge(o,n,~allocate(patient8,203b))`
nurse to all: this bed is in a room that has many beds, for Patient8 we need the smallest room with the fewest occupied beds;
`8. justify(n,*,[defeasible_rule(~allocate(patient8,203b),[large(203), in_room(203b,203)])[as(nurse_statement)], defeasible_rule(allocate(patient8,B),[in_room(B,R),smallest(R) [fewest_occupants(R)]])[as(nurse_restriction)]])`(desiderata (2))
{assistant: `des(n,allocate(patient8,B)[female(B),surgical(B),in_room(B,R), smallest(R)[fewest_occupants(R)]]), defeasible_rule(allocate(patient8,B),[in_room(B,R),smallest(R) [fewest_occupants(R)]])[as(nurse_restriction)]`

operator to assistant: how about allocating Patient8 to the bed that was freed yesterday by Patient6;
`9. assert(o,a,allocate(patient8,B)[allocated(patient6,B,TI,TF), within_time(yesterday,TI,TF)])`
{assistant: `des(n,allocate(patient8,B)[female(B),surgical(B), in_room(B,R),smallest(R)[fewest_occupants(R)]]), defeasible_rule(allocate(patient8,B),[in_room(B,R), smallest(R)[fewest_occupants(R)]])[as(nurse_restriction)], des(o,allocate(patient8,B)[allocated(patient6,B,TI,TF), within_time(yesterday,TI,TF)]), bel(d,allocated(P,B,TI,TF))`} (desiderata (3 and 6))
assistant to database: which bed did Patient6 free yesterday?
`10. question(a,d,allocated(patient6,B,TI,TF) [within_time(yesterday,TI,TF)])`
database to assistant: Patient6 was allocated to bed 202b;
`11. assert(d,a,allocated(patient6,202b,t1,t2))`

assistant to ontology: Is bed 202b suitable to Patient8?
`12. question(a,l,question(suitable(202b,patient8)))`
ontology to assistant: No, it is not.
`13. assert(l,a,~suitable(202b,patient8))`
assistant to operator: the bed freed by patient Patient6 is 202b; it is not suitable for patient Patient8;
`14. assert(a,o,answer(allocated(patient6,202b,t1,t2)`
`15. assert(a,o,answer(~suitable(202b,patient8))`
operator to assistant: why not?
`16. challenge(o,a,~suitable(202b,patient8))`
entering an ontological subdialogue using OAsdlg2
`17. ontoargsubdlg(a,*,~suitable(202b,patient8))`
assistant to ontology: Explain why bed 202b is not suitable to Patient8.
`17a. question(a,l,explain(~suitable(202b,patient8)))`
ontology to assistant: Patient Patient8 is of the age group Adult, and bed 202b is of the age group Adolescent, which is different from Adult. So bed 202b is unsuitable for patient Patient8.
```
17b. assert(l,a,defeasible_rule(
~suitable(202b,patient8),[patient(patient8),
hospital_bed(202b),is_of_the_age_group(patient8,adult),
bed_is_of_the_age_group(202b,adolescent),
differentFrom(adult,adolescent)])[as(nSbyAG)])
```
assistant to all: Patient Patient8 is of the age group Adult, and bed 202b is of the age group Adolescent, which is different from Adult. So bed 202b is unsuitable for patient Patient8. `17c. assert(a,*,answer(defeasible_rule(`
```
~suitable(202b,patient8),[patient(patient8),
hospital_bed(202b),is_of_the_age_group(patient8,
adult),bed_is_of_the_age_group(202b,adolescent),
differentFrom(adult,adolescent)])[as(nSbyAG)])
```
operator to all: Why do you think bed 202b is of the age group Adolescent?
```
17d. challenge(o,*,
bed_is_of_the_age_group(202b,adolescent))
```
assistant to ontology: Explain why bed 202b is of the age group Adolescent.
```
17e. question(a,l,explain(bed_is_of_the_age_group(
202b,adolescent)))
```
ontology to assistant: Patient Patient5 is of the age group Adolescent and is in the same room as the bed 202b
```
17f. assert(l,a,defeasible_rule(
bed_is_of_the_age_group(202b,adolescent),
[patient(patient5),is_of_the_age_group(patient5,
adolescent),hospital_bed(202a),
```

occupy_one(patient5,202a),is_in(202a,202),
hospital_bed(202b),is_in(202b,202)])[as(aGbyPinRoom)])
assistant to all: Patient Patient5 is of the age group Adolescent and is in the same room as the bed 202b
17g. justify(a,*,answer(defeasible_rule(
bed_is_of_the_age_group(202b,adolescent),
[patient(patient5),is_of_the_age_group(patient5,
adolescent),hospital_bed(202a),
occupy_one(patient5,202a),is_in(202a,202),
hospital_bed(202b),is_in(202b,202)])[as(aGbyPinRoom)])
nurse to all: we can make an exception in this case, they can stay in the same room provided they are of the same gender and same type of care;
17h. assert(n,*,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])
{assistant: des(n,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])}
assistant to ontology: Nurse made an exception; Is bed 202b suitable to Patient8 now?
17i. assert(a,l,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])
22a. question(a,l,question(suitable(202b,patient8)))
ontology to assistant: No, it is not.
17j. assert(l,a,~suitable(202b,patient8))
assistant to all: that room is still not suitable;
17k. assert(a,*,~suitable(202b,patient8))
operator to all: why not?
17l. challenge(o,*,~suitable(202b,patient8))
assistant to ontology: Explain why bed 202b is still not suitable for patient Patient8
17m. question(a,l,explain(~suitable(202b,patient8)))
ontology to assistant: Patient Patient8 is in Intensive care, and bed 202b is for Minimal care, which is different from Intensive care. So bed 202b is unsuitable for patient Patient8
17n. assert(l,a,defeasible_rule(
is_unsuitable_for(202b,patient8),[patient(patient8),
hospital_bed(202b),is_care(patient8,intensive),
bed_is_care(202b,minimal),
differentFrom(intensive,minimal)])[as(nSbyCare)]

assistant to all: Patient Patient8 is in Intensive care, and bed 202b is for Minimal care, which is different from Intensive care. so bed 202b is unsuitable for patient Patient8
17o. justify(a,∗,defeasible_rule(
is_unsuitable_for(202b,patient8),[patient(patient8),
hospital_bed(202b),is_care(patient8,intensive),
bed_is_care(202b,minimal),
differentFrom(intensive,minimal)])[as(nSbyCare)]
... **NB:** We suppressed messages from all agents accepting; ~suitable(202b,patient8).
closing ontological subdialogue in agreement that ~suitable(202b,patient8) using rule CloseOAsdlg1 as all agree with that content, the one used to open the subdialogue;
17p. closesubdlg(a,∗,~suitable(202b,patient8))
{assistant: des(o,allocate(patient8,B)[female(B),surgical(B)])}
assistant to operator: Would you like me to try to suggest another bed for Patient8?
18. question(a,o,des(o,suggestion(B,patient8)
[suitable(B,patient8)]))(desiderata (4))
operator to assistant: yes, please!
19. assert(o,a,des(o,suggestion(B,patient8)
[suitable(B,patient8)]))
{assistant: des(n,defeasible_rule(allocate(patient8,B),
[in_room(B,R),smallest(R)[fewest_occupants(R)]]]) [as(nurse_restriction)]),
des(o,suggestion(B,patient8)[suitable(B,patient8)]) (desiderata (3))

assistant to optimiser: generate an allocation suggestion to patient Patient8 considering the nurse restriction.
20. question(a,p,suggestion(B,patient8)[suitable(B,
patient8),defeasible_rule(C,R)[8][as(nurse_restriction)]])
optimiser to assistant: I can not generate a suggestion to patient Patient8 considering this restriction.
21. assert(p,a,~suggestion(B,patient8)
[defeasible_rule(C,R]])[as(nurse_restriction)]]])
{assistant:des(n,defeasible_rule(suitable(B,patient8),
[patient(patient8),bed(B),is_of_the_gender(patient8,G),
bed_is_of_gender(B,G),is_of_care(patient8,C),
bed_is_of_care(B,C)])[as(nurse_exception)])}

assistant to all: can I use the exception made by nurse?
22. question(a,∗,des(o,suggestion(B,patient8)
[suitable(B,patient8),
defeasible_rule(Ce,Re)[as(nurse_exception)]]))(desiderata (5))
nurse to all: yes, you can.

[8]We use the abbreviation C to represent the rule's conclusion and R to represent the body of the rule described just before.

23. assert(n,*,des(o,suggestion(B,patient8)
[suitable(B,patient8),
defeasible_rule(Ce,Re)[as(nurse_exception)]]))
assistant to optimiser: Generate a suggestion considering the nurse restriction and the nurse exception.
24. question(a,p,suggestion(B,patient8)[suitable(B,
patient8),defeasible_rule(Ce,Re)[as(nurse_exception)],
defeasible_rule(Cr,Rr)[as(nurse_restriction)]])
optimiser to assistant: Considering this restrictions and exceptions I recommend bed 201a for Patient8
25. assert(p,a,suggestion(201a,patient8)
[suitable(201a,patient8)])
{assistant: suitable(202a,patient8)}
assistant to all: Considering the exception made by nurse I suggest allocating Patient8 to bed 201a
26. assert(a,*,suggestion(201a,patient8)
[suitable(201a,patient8),
[defeasible_rule(Ce,Re)[as(nurse_exception)]])
operator to assistant: ok, please book bed 201a for Patient8 who will leave the operation room, not before 19:00 nor after 20:30;
27. assert(o,a,booked(201a, patient8, 19:00h, 20:30h))(desiderata (2))
{assistant: des(o,booked(201a, patient8, 19:00h, 20:30h))}
assistant to database: book bed 201a to patient Patient8 from 19:00H and keep the booking until 20:30H;
28. assert(a,d, booked(201a, patient8, 19:00h, 20:30h))
database to assistant: Ok, bed 201a is now booked;
29. accept(d,a,booked(201a,patient8,19:00h,20:30h))
{assistant: booked(201a, patient8, 19:00h, 20:30h)}
assistant to operator: booking done!
30. assert(a,o,booked(201a,patient8,19:00h,20:30h))
operator to assistant: Ok!
31. accept(o,a,booked(201a,patient8,19:00h,20:30h))

Based on Cohen's desiderata for task-oriented dialogue systems and seeking to offer hospital professionals a dialogue as presented above, we loaded our system with real data from the hospital. Then we asked two bed-allocation specialists, who work at that hospital, to evaluate it. Our main objective was to verify if the chatbot's performance matches the points raised by Cohen's desiderata. Next, we describe the test setup and the results obtained.

5.5.2.1 Evaluation Procedure

We received a data load from the hospital that contained patients, beds with their status, bed requests, infections, type of request, type of insurance, coverage, and speciality. During testing, we used 80 beds, 45 inpatients, 42 bed requests, and 65 infections (possibly more than one per patient). Of the 80 beds, there were vacant, occupied by a patient, blocked by infection, maintenance, and cleaning status. We create an ontology with the following settings: Classes (20), Object Properties (29), Data Properties (2), Individuals (22), Datatypes (1), and SWRL Rules (12).

We intentionally invited two bed allocation specialists who work at the hospital. They were selected due to their experience in bed allocation and their knowledge of hospital data. Table 5.5 presents the specialists' demography.

Table 5.5: Specialists demography

Id	**Gender**	**Job occupation**	**How long in this job?**	**Experience in bed allocation**
S1	M	Research center coordinator	1 year	6 years
S2	F	Access management coordinator	6 months	22 years

The evaluation was performed in person at the hospital and simultaneously with the two specialists. After the test, the experts answered an online questionnaire. We created a questionnaire with eight closed questions following the 5-point Likert scale to evaluate Cohen's desiderata (Table 5.6) and an open question to suggestions. During the test, we took notes on what the specialists said. Also, we generate a system log with all requests made, as shown in Apendix C.

Table 5.6: Questions to evaluate Cohen's desiderata

Id	**Question**
Q1	Are the bed allocation rules consistent with those practised in the hospital?
Q2	Are the answers given by the chatbot easily understandable?
Q3	Can the chatbot understand the meaning of user statements, including restrictions with two superlative expressions, one embedded in the other? For example, what is the smallest room with the fewest patients allocated?
Q4	Is the chatbot able to understand the user's intent even if this is not explicitly informed?
Q5	When asked if a bed is suitable for a patient, is the chatbot able to respond and still explain in an easily understandable way how it reached that conclusion?
Q6	Is the chatbot capable of chatting with more than one user at the same time, taking into account the requests and restrictions of all users?
Q7	Does the chatbot reason about what the user is trying to do?
Q8	Can the chatbot understand that another person may have the necessary information/authorization to resolve a situation and is able to request that this person be asked?

We started the test by explaining the system and the objective of the evaluation. Then, we show the interface (Figure 5.4) where they could see the patients' and beds' data and provide users with examples of questions they could ask the chatbot. After signing the consent form, the specialists started the test.

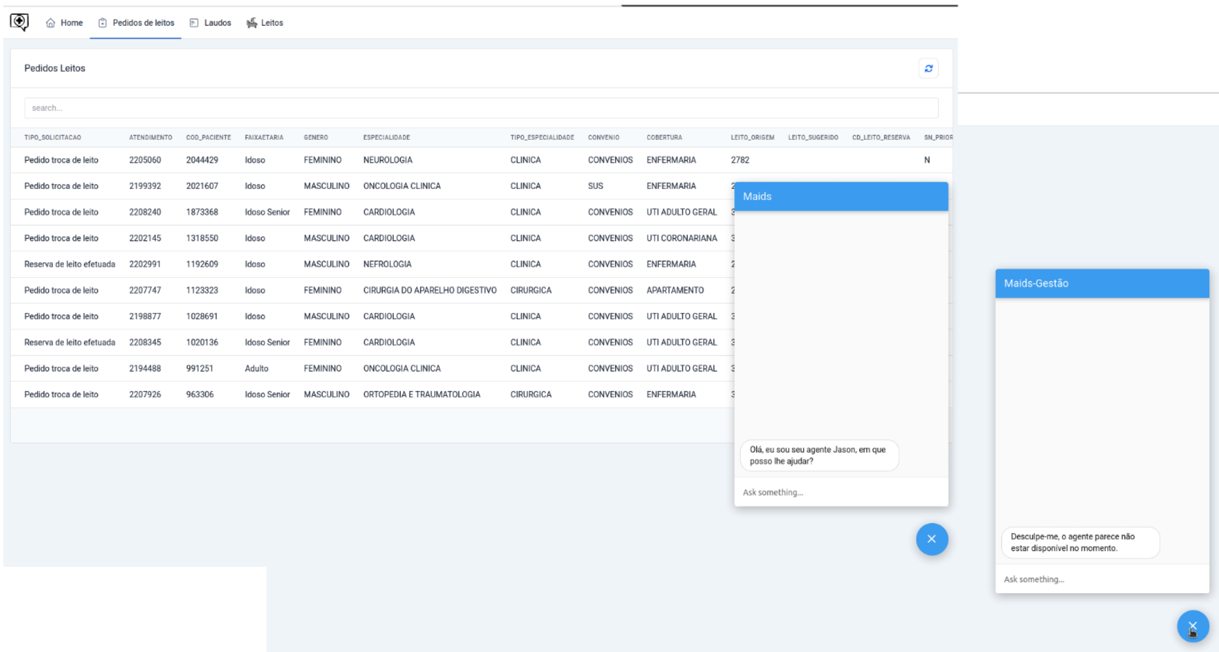

Figure 5.4: MAIDS testing interface

One of the evaluators performed the test remotely via video call and the other in person. Both evaluators requested that questions be asked to the chatbot, and the questions were typed in by us. After viewing the response provided by the chatbot, the evaluators decided together what would be the next question to be asked and informed us so that it could be typed into the chatbot. After a series of questions and answers, the evaluators declared themselves satisfied and started answering the questionnaire.

5.5.2.2 Results

Regarding the evaluation of the chatbot by the specialists, Figure 5.5 shows an excerpt from the conversation log where the operator asks if he can put a patient in a room, which is not possible, but the operator wants it anyway. Then, the nurse in charge is asked to make an exception, so after that, it is possible to carry out the allocation. This shows that simultaneous conversations with the chatbot are possible and that it considers requests from both users. After that, we can see in Figure 5.6 that the chatbot can handle superlative expressions; the operator asked which is the smallest room with fewer patients and which is the biggest room with more patients allocated. Room 118J and 759 were answered, respectively.

Agent operator asks to allocate patient 2044429 to bed 3132
Answering to chatbot: **First I need to check if this bed is suitable for this patient, is that ok?**
Answering to chatbot: **No, this bed is not suitable.**
Chatbot of operator is requesting explanation about suitability of 3132 bed to 2044429
Answering to chatbot: **Bed 3132 is not suitable for patient 2044429 as patient coverage is WARD and this bed is GENERAL ADULT ICU.**
Chatbot of operator is requesting to allocate anyway.
Answering to chatbot: **Sorry, you are not authorized to make this allocation, but you can ask the manager to make an exception**
Chatbot of nurse is requesting to register an exception: patient: 2044429, bed: 3132
Agent nurse asks to register exception on patient 2044429 and bed 3132
Answering to chatbot: **Ok I am logging the requested exception**
Chatbot of operator is requesting to verify suitability: patient: 2044429, bed: 3132
Answering to chatbot: **Yes, this bed is suitable.**
Chatbot of operator is requesting explanation about suitability of 3132 bed to 2044429
Answering to chatbot: **Bed 3132 is suitable for patient 2044429 as the manager made an exception for this case.**

Figure 5.5: Simultaneous conversation and restriction

Chatbot of operator is asking about the smallest room with fewer patients allocated.
Answering to chatbot: **The smallest room with the fewest occupants is 118J**
Chatbot of operator is asking about the biggest room with most patients allocated.
Answering to chatbot: **The biggest room with most occupants is 759**

Figure 5.6: Superlative expressions

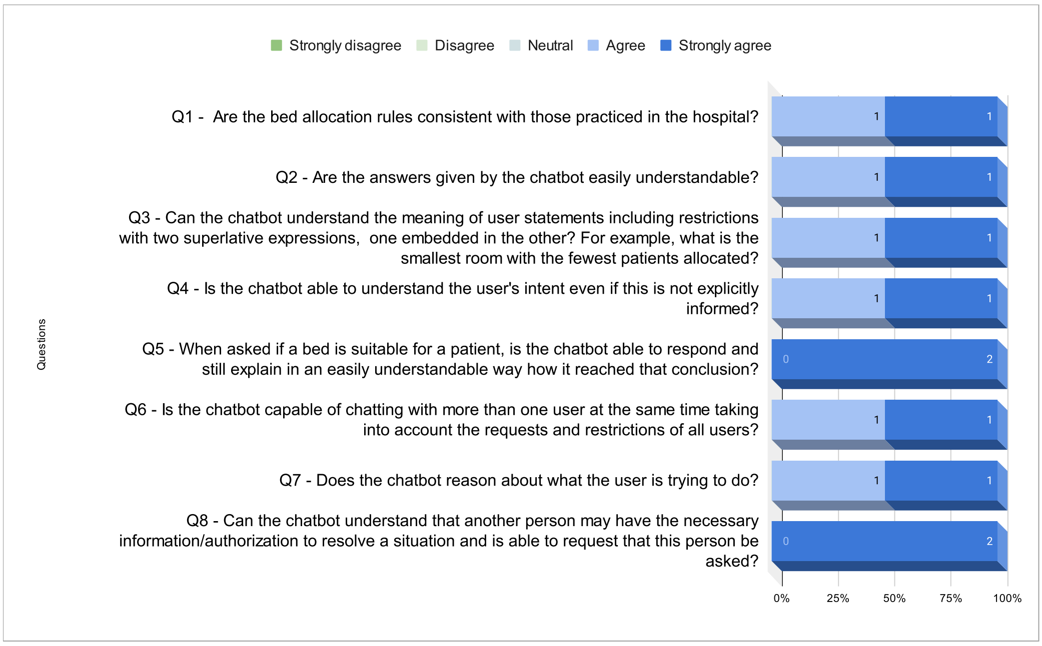

Figure 5.7: Evaluation result

The results of the questionnaire indicated that specialists strongly agree or agree that the system respect Cohen's desiderata, according to Figure 5.7. During the test, they suggested it would be very helpful if the system had voice input. Also, it would be useful if, when asking the chatbot about free beds for women, for example, the MAIDS brings a list of free beds that can be used. Today MAIDS brings one.

FINAL REMARKS

This chapter presented some tools developed specifically for the bed allocation domain we use as a case study, such as a domain ontology, a Java-based PDDL plan validator, an optimiser using the GLPSol solver of GLPK, and two RML properties to perform runtime verification. In addition, we describe the evaluations made with professionals responsible for the bed allocation at Hospital São Lucas da PUCRS in Brazil, where the functionalities offered by the system developed using the MAIDS framework and the expressiveness of this system based on Cohen's desiderata were evaluated separately. The contents of this chapter have been published in [ECG+19, ECPB21].

Chapter 6

Related Work

The only work that supports agents arguing about OWL ontologies specifically, to the best of our knowledge, appeared in [MB08]. However, that work was not formalised in the context of an agent programming language, and did not support ToM, nor the structured dialogue approach we introduced in this thesis. Furthermore, that framework does not seem to have been further developed and does not seem to be available for download, so it does not support the development of practical dialogue systems like ours. In fact, we are unaware of any practical agent framework that supports all the features of dialogue systems supported by our framework.

There is much work on allowing for defeasibility in description logic and OWL [GG19, AW03], but this is also distant from our work in that it does not provide practical support for agent programming with argumentation-based dialogues. Although there is work on nested dialogues [BH07], the possibility to digress about ontological and ToM issues in subdialogues as put forward in this thesis is completely original.

Another strand of work in argumentation to mention here is on using automated planning techniques to support an agent's strategy in taking part in dialogue games [BCH17]. In the context of XAI, as described in [ANCF19], there is little work addressing the issues of multi-agent explainability, personalisation of explanation, and context awareness. Also in the explainable AI context, Arg2P, for example, is a logic-based argumentation framework for defeasible reasoning and agent conversation based on a modular architecture allowing for system transparency and ease of extension. Unlike MAIDS, which is implemented in AOPL, the Arg2P is an $ASPIC^{+}$-like implementation for structured argumentation.

Regarding works that seek to use argumentation to make dialogue systems more explainable to the user, we present some works in Table 6.1.

Table 6.1: Related work summary

Author	Argumentation Techniques	XAI Techniques	Multiple domains	Natural Language Interaction
[TYZ12]	Fuzzy Cognitive Map based on argumentation model	Not cited	No	No
[GMBE+13]	ASPIC rules and argumentation schemes	Argumentation graph	No	No
[KÕ15]	Naive theory of reasoning	Not cited	No	Yes
[TPW+16]	Graphical representation of arguments structure, argumentation-based reasoning, and argumentation schemes	Not cited	No	No
[CQTF17]	Pragmatic argumentation	Not cited	No	No
[BJG+17]	Argumentation scheme and ontology	Not cited	No	No
[DB17]	Argumentation theory	Not cited	No	No
[ERS+18]	Argumentation-based dialogue	Not cited	No	No
[RWP+18]	Argument game	Not cited	No	No
[SIH+18]	Argumentation scheme	Not cited	No	Yes
[RBR+19]	Regression, agglomerative clustering, similarity algorithm (UMBC STS score)	Not cited	No	No
[KCB+19]	Argumentation scheme and attack schemes	Not cited	No	Yes
[P+20]	Crowd sourced argument graph	Not cited	No	Yes
[WJR+20]	Argument annotation scheme and bipolar weighted argument graphs (BWAGS)	Not cited	No	No
MAIDS (our approach)	Argumentation scheme and argumentation-based dialogues	Argumentation scheme and ontological reasoning	Yes	Yes

Tao et al. [TYZ12] used a Fuzzy Cognitive Map based on argumentation model to construct an intelligent tutoring system, with an intelligent agent to conduct argumentative dialogues helping children learn ecosystems and adults to gain knowledge on diabetes risk factors. The user interacts with the chatbot using pre-defined buttons. The system has not yet been evaluated.

Grando et al. [GMBE+13] introduce an argument-based system for enhancing human-computer dialogues in the medical domain, more specifically, in medical training. The user interacts with the chatbot using buttons. The system has not yet been evaluated.

Koit and Õim [KÕ15] used the naive theory of reasoning to model argumentation in agreement negotiation processes. They created an experimental dialogue system implemented in Java where the user can both choose their sentences from a menu or put them in free text, which will be interpreted by the computer using keywords or phrases to classify the user texts semantically. In both, the computer uses a list of ready-made sentences about business trips and vegetarianism in Estonian [KÕ15]. Six voluntary users have carried out some experiments with the dialogue system. However, the generated dialogues are not quite coherent because the computer uses predefined sentences.

Toniolo et al. [TPW+16] developed Moira (Mobile Intelligence Reporting Agent), a Controlled Natural Language (CNL) conversational agent, that uses a conversational model based on the syntax and semantics of the ITA Controlled English [PBPP14]. They used the CISpaces (Collaborative Intelligence Spaces) toolkit based on a graphical representation of structured arguments. They analysed through argumentation-based reasoning and argumentation schemes to support the identification of plausible hypotheses. CISpaces structure and share analyses of conflicting information through argumentation techniques and maintains records of the provenance information. While Moira captures sensed information by supporting queries and intelligence provision from a wide set of different sources, such as physical sensors and information systems, social media and human sources. They demonstrated a scenario showing the support offered by Moira and CISpaces to an intelligence coalition while gathering information about and responding to the attack on Kish.

Cheng et al. [CQTF17] created a recommendation system using pragmatic argumentation to check if this has the potential to affect the decision making of the elderly and help him/her pursue a healthier life. The system has only recommendations, it did not have free conversation. The system was evaluated by 21 volunteers who interacted with the robot.

Baskar et al. [BJG+17] developed a coaching to improve health and well-being using multi-agent system and argumentation. The user interacts with the chatbot using pre-defined buttons. The system has not yet been evaluated.

Dignum and Floris [DB17] discussed an approach to dialogue management using chatbots in combination with social practices and argumentation theory. Their future system provides for the use of an avatar that expresses emotions and the user will be able to utterances with natural language. The system has not yet been evaluated.

Essers et al. [ERS+18] created a prototype called POSTURE (Pressure Offloading Support Technologies for Ulcer REduction). This prototype help patients suffering from Diabetic Foot Ulcer (DFU) self-manage their treatment. This system receives data from commercial wellness sensors. The sensor data is fed into a data-backed decision support tool that interacts with users via an intelligent agent, using a chatbot style interface, underpinned by computational argumentation-based dialogue. The user is not able to chat freely with the chatbot. Regarding evaluation, 10 users interacted with the prototype via tablet interface and fictitious data, then answer a questionnaire about the idea of such a system.

Rach et al. [RWP$^+$18] used Charamel tool[1] to create the avatar EVA, a multimodal argumentative dialogue system that uses Nuance text-to-speech and all Amazon Polly Voices. They use an argument game based on the formalism of Prakken [Pra00] to model the interaction between the user and the system. The avatar uses a pre-defined template of mimics and gestures that will be replaced by customized signals in the future. It presents arguments about marriage and supports them. The system has not yet been evaluated.

Sakai et al. [SIH$^+$18] present a chat interface that generated dialogue text based on argumentation structures. They created a model with a graph structure in which nodes represent premises and edges represent the relationship between nodes. Each node has a natural language statement representing the content of its premise. If the logical connection is based on an argumentation scheme, the scheme name is represented on the arcs. More than 30 annotators were recruited to create the argumentation structures. Since they created the argument structures manually, it is quite costly to expand and create new structures for other topics. They evaluated the dialogue system with 19 participants who read ten dialogue texts that were randomly ordered and answered a questionnaire.

Rakshit et al. [RBR$^+$19] created a chatbot called Debbie that uses retrieval from existing conversations to argue with users. In Debbie, the user picks a button with three topics - death penalty, gay marriage, and gun control and inform about their position (for or against). Then, Debbie generate arguments against that position using a similarity algorithm (UMBC STS score) to retrieve a ranked list of the most appropriate counter-arguments, i.e., arguments opposing the user's stance. They evaluated the average response times for each retrieval method and the quality of the responses testing three sentences per stance per topic.

Kokciyan et al. [KCB$^+$19] developed a system called CONSULT, which is a collaborative decision-support tool to help patients suffering from chronic diseases to self-manage their treatment plans. The system has a conversation interface with the user, in which the user can make utterances in natural language. The system has not yet been evaluated.

Prakken et al. [P$^+$20] developed a chatbot that was deployed on Facebook via the Messenger Send/Receive API to engage and persuade the user to accept university fees should be kept. They evaluated the chatbot with 50 participants. The participants interacted with the chatbot. Before using the chatbot, participants answered a questionnaire on google forms indicating whether they strongly disagreed, disagreed, neutral, agreed or strongly agreed that university fees should be kept. After using the chatbot, users answered another questionnaire on google forms informing about the experience and "How much do you agree that fees in the UK should be kept as they are? (Strongly disagree - strongly agree)".

Weber et al. [WJR$^+$20] presented Gloria avatar, which can perform lip-sync speech output using the Nuance text-to-speech along with the Amazon Polly voices. They used an argument structure based on the argument annotation scheme. Their study includes three types

[1] `https://www.charamel.com/competence/avatare`

of argument components (Major Claim, Claim, and Premise) and two different directed relations (support and attack) between them. They infer the argumentation structure using a hotel reviews dataset annotated through an argument mining approach. The annotated sentences serve as a template for the system's natural language generation. They determined the polarity of each review by making a comparison between positive and negative phrases annotated. The prediction model computes the user's preferences based on bipolar weighted argument graphs (BWAGS) using linear Euler-based restricted semantics. Gloria talks about a topic by giving arguments either for or against the topic. The agent can try to influence the user using appropriate emotions (angry, happy, and sad). After each argument from the agent, the user has three button options to give their feedback on the argument (convincing, neutral, not convincing).

Comparing those works with MAIDS (Table 6.1) we can see we did not find research which can be applied to multiple domains. None of the research on bed allocation with decision-support characteristics and explainability based on argumentation theory was found. Also, only in [KÕ15, SIH+18, KCB+19, P+20] is allowed free user interaction with the chatbot, i.e., where the user can interact in natural language with the agent.

Chapter 7

Conclusions

Our main objective in this thesis was *to investigate how argumentation theory and ontology techniques can be used together with reasoning about intentions to build complex natural language dialogues to support humans in decision making*. The thesis puts forward a framework for developing explainable multi-agent systems. Its potential is demonstrated by a case study on human-agent dialogues on hospital bed allocation. We achieved that goal through the following specific objectives:

- Design and formalise an approach to argumentation-based dialogues and ontological reasoning. In Chapter 3 we presented and formalised an approach that supports the development of dialogue systems based on BDI agents to assist humans in decision making. Using this approach, agents have a multi-part belief base and support for having a structured dialogue where the main line of argumentation is based on the argumentation schemes knowledge component. Further, it can lead to subdialogues when ontological or ToM issues must be resolved [EPV$^+$].
- Construct an architecture/framework for developing explainable systems and implement all the necessary components of the architecture. In Chapter 4, we present our framework for developing multi-agent intentional dialogue systems. We emphasize that the MAIDS framework can be used in its entirety, or it is also possible to use only the modules (Dial4JaCa, Onto4JaCa, RV4JaCa) that match the requirements of each system under development. We have some publications related to this part of the work ([Eng20, EDK$^+$21b, EDK$^+$21a, EDP$^+$22, PEB21, FPE$^+$22, EFP$^+$22, OEK$^+$22]).
- Apply the proposed formal model to a real-world domain and problem, such as healthcare, and fully implement a dialogue system based on that formalisation using the framework. In Chapter 5, a case study in the hospital bed allocation domain is shown. We implemented some tools related to the bed allocation domain, such as a domain on-

tology published in [ECG+19], a Java-based PDDL plan validator, and an optimiser using the linear programming solver of GLPK [ECPB21]. Furthermore, using our framework, we built an explainable dialogue system to support decision making for hospital bed allocation, which used real data from a hospital.

- Evaluate the approach with domain experts based on Cohen's desiderata for task-oriented dialogue systems. As shown in Section 5.5, staff from PUCRS's Hospital São Lucas, in Porto Alegre, Brazil, supported us in evaluating our system. We performed two evaluations with professionals responsible for bed allocation in that hospital. The first one was to evaluate the system functionalities for hospital bed allocation, while the second one was to evaluate the expressiveness of our framework based on Cohen's desiderata for task-oriented dialogue systems.

7.1 Summary of Results and Discussion

The use of artificial intelligence systems in our daily lives is becoming mature and ubiquitous, resulting in the growing availability of systems where agents and humans work together [RR19], from which emerge new concepts such as Hybrid Intelligence [ABdR+20]. However, there are many challenges related to how humans and agents will interact in those systems and how agents will explain their decisions and internal mental states so that they become more transparent and trustworthy [Gun17, GSC+19b].

In this thesis, we contribute in this direction by providing tools for the practical development of explainable intelligent systems as well as systems supporting hybrid intelligence. Aiming at communication between humans and software agents, we built Dial4JaCa, which acts as a bridge between multi-agent systems developed with JaCaMo and natural language processing platforms such as Dialogflow and Rasa. This way, we endow intelligent agents to communicate in natural language with human users. Onto4JaCa, in turn, was developed to enable agents to use, during their reasoning process, the information contained in ontologies as well as inferred by semantic reasoners. In addition, we built RV4JaCa, capable of observing all messages exchanged by agents during runtime and forwarding them to a formal monitor in order to verify properties of interest and, if necessary, interfere with the execution of the system to recover from or prevent failures. These three modules form the basis of MAIDS, our framework for developing multi-agent intentional dialogue systems, which can be used, together with domain-specific agents, in different domains. As our framework is modular, it can be used in its entirety or just the modules that fulfil the requirements of each system to be developed. Our work also includes the formalisation of a novel dialogue-subdialogue structure with which we can address ontological or theory-of-mind issues and later return to the main subject of the dialogue.

In order to evaluate the practical applicability of our framework, we have developed a multi-agent system that supports hospital bed allocation. In addition to the modules of the MAIDS framework, some specific tools for this domain were also developed, such as a domain ontology, a PDDL plan validator, an optimiser, and two RML properties to be verified during runtime. The developed system was evaluated by professionals specialised in bed allocation at Hospital São Lucas – PUCRS, who agreed to support us in evaluating our approach. Two separate evaluations were carried out, the first one, with synthetic data, in order to check whether the functionalities offered by our system fulfilled the hospital's needs. The second one, using real data from the hospital, aimed to evaluate the expressiveness of a system developed with the MAIDS framework based on Cohen's desiderata for task-oriented dialogue systems. As a result of the evaluations, we were able to conclude that, according to the evaluators, our system addresses all the elements of the desiderata.

Although much work remains to be done, as discussed in Section 7.3, in its current state, our framework already contributes towards increased sophistication in explainable AI, hybrid intelligence, and human-agent dialogue systems.

7.2 Thesis publications

We have published the following papers related to the research presented in this thesis, in particular the results described above:

- "Towards an ontology to support decision making in hospital bed allocation" [ECG+19];
- "Conversational Agents Based on Argumentation Theory and Ontologies" [Eng20];
- "Dial4JaCa – A Communication Interface Between Multi-agent Systems and Chatbots" [EDK+21b];
- "Dial4JaCa – A Demonstration" [EDK+21a];
- "A Conversational Agent to Support Hospital Bed Allocation" [ECPB21];
- "Argumentation as a Method for Explainable AI: A Systematic Literature Review" [EDP+22];
- "Engineering Explainable Agents: An Argumentation-Based Approach" [PEB21];
- "Explaining Semantic Reasoning using Argumentation" [FPE+22];
- "RV4JaCa - Runtime Verification for Multi-Agent Systems" [EFP+22];
- "Multi-Agent Interaction to Assist Visually-Impaired and Elderly People" [OEK+22];
- "MAIDS — a Framework for the Development of Multi-Agent Intentional Dialogue Systems" [EPV+].

7.3 Future Work

During the development of this thesis, it was possible to identify future directions arising from the results and their limitations, which may allow future work to increase knowledge and its application on this topic. For example, one of the directions to follow would be the use of more complex ontological subdialogues than those presented in this thesis, in addition to the practical use of ToM subdialogues, since although we formalise them here, we do not use subdialogues based on the theory of mind in practice in our case study.

In addition, there is a lack of argumentation schemes aimed at reasoning patterns to discuss theory of mind. So, efforts should be made to identify schemes that justify or attempt to invalidate an inference from a mental attitude. Possibly there are general schemes, but new argumentation schemes may be necessary depending on the application domain. Another venue for future work is related to the complexity of arguments over time. It is necessary to consider that information can be true at a given moment but cease to be true over time.

We also aim to apply automated planning techniques to support an agent's strategy in taking part in dialogue games [BCH17] to decide when to move to subdialogues (currently, we used a simple strategy for the case study, one that moves to subdialogues as soon as possible). Future work also includes allowing a subset of the agents entering a subdialogue as well as other agents that are not involved in the main dialogue but may have some information needed to reach a subdialogue conclusion. Further developing the applications so they also use the ToM subdialogues, and experimenting with our framework to develop dialogue systems in other hospital management domains besides bed allocation, are also possible future work.

Finally, it is also possible to use the MAIDS framework or some of its modules in building explainable multi-agent systems in other domains, such as Law and IoT (Internet of Things), to help people in decision making, bringing more areas of application to the direction of hybrid intelligence.

Appendix A

Bed-Allocation Plan Validation Report

PDDL Plan Validator
Validation Report

1 Domain

Domain: hospital
Problem: aloc

1.1 Files

Domain file: test\domain.pddl
Problem file: test\problem.pddl
Plan file: test\plan.pddl

2 Plan

1. (**aloc-Oncologia** 34345454354 100a genero–Masculino genero–Masculino)
2. (**aloc-Oncologia** 93092302930 100c genero–Masculino genero–Masculino)
3. (**aloc-Oncologia** 09090909091 100b genero–Masculino genero–Masculino)
4. (**aloc-Medicina-Interna** 23433454353 112a genero–Feminino genero–Feminino)
5. (**aloc-Medicina-Interna** 34343434324 112d genero–Feminino genero–Feminino)
6. (**aloc-Neurologia** 23266656502 113b genero–Masculino genero–Masculino)
7. (**aloc-Cardiologia** 32323233232 114a genero–Masculino genero–Masculino)
8. (**aloc-Neurologia** 32135131355 115b genero–Feminino genero–Feminino)
9. (**aloc-Cardiologia** 23102103133 116b genero–Feminino genero–Feminino)
10. (**aloc-Oncologia** 54532513216 117e genero–Feminino genero–Feminino)

3 Validation

1. (**aloc-Oncologia** 34345454354 100a genero–Masculino genero–Masculino)
   - +(*alocado* 34345454354)
   - +(*ocupado* 100a)
   - +(*in* 34345454354 100a)
2. (**aloc-Oncologia** 93092302930 100c genero–Masculino genero–Masculino)
   - +(*alocado* 93092302930)
   - +(*ocupado* 100c)
   - +(*in* 93092302930 100c)
3. (**aloc-Oncologia** 09090909091 100b genero–Masculino genero–Masculino)
   - +(*alocado* 09090909091)

- +(*ocupado* 100b)
- +(*in* 09090909091 100b)

4. (**aloc-Medicina-Interna** 23433454353 112a genero–Feminino genero–Feminino)
   - +(*alocado* 23433454353)
   - +(*ocupado* 112a)
   - +(*in* 23433454353 112a)
5. (**aloc-Medicina-Interna** 34343434324 112d genero–Feminino genero–Feminino)
   - +(*alocado* 34343434324)
   - +(*ocupado* 112d)
   - +(*in* 34343434324 112d)
6. (**aloc-Neurologia** 23266656502 113b genero–Masculino genero–Masculino)
   - +(*alocado* 23266656502)
   - +(*ocupado* 113b)
   - +(*in* 23266656502 113b)
7. (**aloc-Cardiologia** 32323233232 114a genero–Masculino genero–Masculino)
   - +(*alocado* 32323233232)
   - +(*ocupado* 114a)
   - +(*in* 32323233232 114a)
8. (**aloc-Neurologia** 32135131355 115b genero–Feminino genero–Feminino)
   - +(*alocado* 32135131355)
   - +(*ocupado* 115b)
   - +(*in* 32135131355 115b)
9. (**aloc-Cardiologia** 23102103133 116b genero–Feminino genero–Feminino)
   - +(*alocado* 23102103133)
   - +(*ocupado* 116b)
   - +(*in* 23102103133 116b)
10. (**aloc-Oncologia** 54532513216 117e genero–Feminino genero–Feminino)
    - +(*alocado* 54532513216)
    - +(*ocupado* 117e)
    - +(*in* 54532513216 117e)

Goal achieved

Appendix B

Evaluation of System Functionalities – Logs

MAS log from specialist S1

```
[NetworkListener] Started listener bound to [{0}]
[HttpServer] [{0}] Started.
[Cartago] Workspace wp created.
[Cartago] getJoinedWorkspaces: [main]
[Cartago] artifact dial4jaca:
br.pucrs.smart.Dial4JaCa.Dial4JaCaArtifact() at wp created.
[Cartago] artifact rv4jaca: br.pucrs.smart.rv4JaCa.RV4JaCaArtifact() at
wp created.
[Cartago] artifact pddl: br.pucrs.smart.validator.ValidatorArtifact() at
wp created.
[Cartago] artifact optimiser:
br.pucrs.smart.optimiser.OptimiserArtifact() at wp created.
[Cartago] artifact postgres: br.pucrs.smart.postgresql.PostgresArtifact()
at wp created.
[OntologyArtifact] Importing ontology from
src/resources/HospitalBedAllocationNoIndividualsand5rules-pt.owl
[OntologyArtifact] Ontology ready!
[Cartago] artifact onto:
br.pucrs.smart.ontology.mas.OntologyArtifact("src/resources/HospitalBedAl
locationNoIndividualsand5rules-pt.owl") at wp created.
[assistant] Assistant agent enabled.
[optimiser] joinned workspace wp
[operator] joinned workspace wp
[validator] joinned workspace wp
[database] joinned workspace wp
[validator] focusing on artifact pddl (at workspace wp) using namespace
default
[operator] focusing on artifact dial4jaca (at workspace wp) using
namespace default
[ontology_specialist] joinned workspace wp
[database] focusing on artifact postgres (at workspace wp) using
namespace default
[optimiser] focusing on artifact optimiser (at workspace wp) using
namespace default
[ontology_specialist] focusing on artifact onto (at workspace wp) using
namespace default
[database] Database specialist agent enabled.
[operator] Communication specialist agent enabled.
[optimiser] Optimiser agent enabled.
[ontology_specialist] focusing on artifact postgres (at workspace wp)
using namespace default
[validator] Validator agent enabled.
[ontology_specialist] Agent ontology_specialist enabled.
[operator] Request received - Get Suggestion from Dialog
[operator] Chatbot of operator suggestions to allocate: ["Marisa da
Costa"]
[assistant] Agent operator requesting suggestion.
[optimiser] Agent assistant requesting optmised suggestion.
[assistant] Result received from agent optimiser
[database] Agent optimiser wants to save the optimiser result
[operator] Answering to chatbot: Eu posso sugerir colocar  o/a paciente
Marisa da Costa no leito 402b. Você gostaria que eu confirmasse essa
alocação?
[optimiser] Resultado da otimização salvo no banco de dados
[operator] Request received - Get Suggestion - no from Dialog
```

[operator] Chatbot of operator is requesting the cancellation of the suggested allocation.
[assistant] Agent operator requesting cancellation of the optmised allocation.
[optimiser] Agent assistant requesting cancellation of the last optmised allocation.
[database] Agent optimiser wants to cancel the last optmised allocation
[optimiser] Agent database is returning about the cancellation of the last optmised allocation.
[assistant] Agent optimiser is answering about the cancellation of the optmised allocation.
[operator] Answering to chatbot: Ok, solicitação concluida sem alocar nenhum paciente.

RV4JaCa log from specialist S1

{"msgId":"mid1","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"getSuggestionByPatient","prop1":{"pos1":"\"Marisa da Costa\""},"isNegated":false}}
{"msgId":"mid2","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"getSuggestionByPatient","prop1":{"pos1":"\"Marisa da Costa\""},"isNegated":false}}
{"msgId":"mid3","isReply":"nirt","performative":"assert","sender":"optimiser","receiver":"assistant","content":{"name":"suggestionByPatient","prop1":{"prop2":{"name":"notAlloc","isNegated":false},"prop1":"\"true\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"prop1":{"prop2":"\"402b\"","prop1":"\"Marisa da Costa\"","name":"alloc","isNegated":false}},"name":"sugestedAllocation","isNegated":false}},"isNegated":false}}
{"msgId":"mid4","isReply":"nirt","performative":"question","sender":"optimiser","receiver":"database","content":{"name":"saveOptimiserResult","isNegated":false}}
{"msgId":"mid5","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Eu posso sugerir colocar o/a paciente Marisa da Costa no leito 402b. Você gostaria que eu confirmasse essa alocação?\"","name":"answer","prop1":{"prop1":{"prop2":{"name":"notAlloc","isNegated":false},"prop1":"\"true\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"prop1":{"prop2":"\"402b\"","prop1":"\"Marisa da Costa\"","name":"alloc","isNegated":false}},"name":"sugestedAllocation","isNegated":false}},"name":"suggestionByPatient","isNegated":false},"isNegated":false}}
{"msgId":"mid6","isReply":"nirt","performative":"assert","sender":"database","receiver":"optimiser","content":{"name":"savedOptimiserResult","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid7","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"cancelAllocByOptimization","isNegated":false}}
{"msgId":"mid8","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"cancelOpAlloc","isNegated":false}}
{"msgId":"mid9","isReply":"nirt","performative":"question","sender":"optimiser","receiver":"database","content":{"name":"cancelOpAlloc","isNegated":false}}
{"msgId":"mid10","isReply":"nirt","performative":"assert","sender":"database","receiver":"optimiser","content":{"name":"cancelOpAllocation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid11","isReply":"nirt","performative":"assert","sender":"optimiser","receiver":"assistant","content":{"name":"cancelOpAllocation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid12","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Ok, solicitação concluida sem alocar nenhum paciente.\"","name":"answer","prop1":{"prop1":"\"Success\"","name":"cancelOpAllocation","isNegated":false},"isNegated":false}}

MAS log from specialist S2

```
[NetworkListener] Started listener bound to [{0}]
[HttpServer] [{0}] Started.
[Cartago] Workspace wp created.
[Cartago] getJoinedWorkspaces: [main]
[Cartago] artifact dial4jaca:
br.pucrs.smart.Dial4JaCa.Dial4JaCaArtifact() at wp created.
[Cartago] artifact rv4jaca: br.pucrs.smart.rv4JaCa.RV4JaCaArtifact() at
wp created.
[Cartago] artifact pddl: br.pucrs.smart.validator.ValidatorArtifact() at
wp created.
[Cartago] artifact optimiser:
br.pucrs.smart.optimiser.OptimiserArtifact() at wp created.
[Cartago] artifact postgres: br.pucrs.smart.postgresql.PostgresArtifact()
at wp created.
[OntologyArtifact] Importing ontology from
src/resources/HospitalBedAllocationNoIndividualsand5rules-pt.owl
[OntologyArtifact] Ontology ready!
[Cartago] artifact onto:
br.pucrs.smart.ontology.mas.OntologyArtifact("src/resources/HospitalBedAl
locationNoIndividualsand5rules-pt.owl") at wp created.
[assistant] Assistant agent enabled.
[optimiser] joinned workspace wp
[optimiser] focusing on artifact optimiser (at workspace wp) using
namespace default
[validator] joinned workspace wp
[validator] focusing on artifact pddl (at workspace wp) using namespace
default
[database] joinned workspace wp
[database] focusing on artifact postgres (at workspace wp) using
namespace default
[operator] joinned workspace wp
[ontology_specialist] joinned workspace wp
[database] Database specialist agent enabled.
[validator] Validator agent enabled.
[optimiser] Optimiser agent enabled.
[operator] focusing on artifact dial4jaca (at workspace wp) using
namespace default
[ontology_specialist] focusing on artifact onto (at workspace wp) using
namespace default
[operator] Communication specialist agent enabled.
[ontology_specialist] focusing on artifact postgres (at workspace wp)
using namespace default
[ontology_specialist] Agent ontology_specialist enabled.
[operator] Request received - Get Suggestion from Dialog
[operator] Chatbot of operator suggestions to allocate: ["Maria da
Silva"]
[assistant] Agent operator requesting suggestion.
[optimiser] Agent assistant requesting optmised suggestion.
[assistant] Result received from agent optimiser
[database] Agent optimiser wants to save the optimiser result
[operator] Answering to chatbot: Eu posso sugerir colocar  o/a paciente
Maria da Silva no leito 103a. Você gostaria que eu confirmasse essa
alocação?
[optimiser] Resultado da otimização salvo no banco de dados
[operator] Request received - Get Suggestion - no from Dialog
```

```
[operator] Chatbot of operator is requesting the cancellation of the
suggested allocation.
[assistant] Agent operator requesting cancellation of the optmised
allocation.
[optimiser] Agent assistant requesting cancellation of the last optmised
allocation.
[database] Agent optimiser wants to cancel the last optmised allocation
[optimiser] Agent database is returning about the cancellation of the
last optmised allocation.
[assistant] Agent optimiser is answering about the cancellation of the
optmised allocation.
[operator] Answering to chatbot: Ok, solicitação concluida sem alocar
nenhum paciente.
[operator] Request received - Get Suggestion from Dialog
[operator] Chatbot of operator suggestions to allocate: ["Maria da
Silva"]
[assistant] Agent operator requesting suggestion.
[optimiser] Agent assistant requesting optmised suggestion.
[assistant] Result received from agent optimiser
[database] Agent optimiser wants to save the optimiser result
[operator] Answering to chatbot: Eu posso sugerir colocar  o/a paciente
Maria da Silva no leito 103a. Você gostaria que eu confirmasse essa
alocação?
[optimiser] Resultado da otimização salvo no banco de dados
[operator] Request received - Get Optimised Allocation from Dialog
[operator] Chatbot of operator is requesting an optimised allocation.
[assistant] Agent operator requesting an optmised allocation.
[optimiser] Agent assistant requesting an optmised allocation.
[optimiser] Calling Optimiser.
[database] Agent optimiser wants to save the optimiser result
[assistant] Result received from agent optimiser
[operator] Answering to chatbot: Gerei uma alocação otimizada, porém não
conseguirei alocar todos os pacientes pois não localizei leitos adequados
para os pacientes Vitor Almeida, e Daniel Souza. Você pode ver minha
sugestão no menu 'Alocação otimizada' aqui ao lado. Você quer que eu
confirme essa alocação?
[optimiser] Resultado da otimização salvo no banco de dados
[operator] Request received - Get Optimised Allocation - yes from Dialog
[operator] Chatbot of operator is requesting the confirmation of the
optimised allocation.
[assistant] Agent operator requesting confirmation of the optmised
allocation.
[optimiser] Agent assistant requesting confirmation of the optmised
allocation.
[database] Agent optimiser wants to allocate patients using optimizer
result
[assistant] Communication error -- no_applicable: Found a goal for which
there is no applicable plan:
+!kqml_received(optimiser,assert,allocatedOpPatients("Erro"),mid29)
[operator] Request received - Get Validation Result from Dialog
[operator] Chatbot of operator is requesting plan validation.
[assistant] Agent operator requesting validation result.
[validator] Agent assistant requesting validation result.
[assistant] Response received from agent validador
[validator] Agent assistant requesting to save result in validation.
[database] Agent validator wants to update the validation result
[operator] Answering to chatbot: O seu plano de alocação de leitos possui
falhas. Houve um erro ao alocar os seguintes pacientes: Vitor Almeida no
```

leito 301b - Pois o leito 301b não é de Tipo de Encaminhamento Agudo, e Especialidade Gastro como é o caso do paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira uma alocação otimizada?
[validator] Agent database is answering about the update.
[assistant] Retorno da validação salvo no banco de dados
[operator] Request received - Get Validation Result - optimize from Dialog
[operator] Chatbot of operator is requesting an optimised allocation.
[assistant] Agent operator requesting an optmised allocation.
[optimiser] Agent assistant requesting an optmised allocation.
[optimiser] Calling Optimiser.
[database] Agent optimiser wants to save the optimiser result
[assistant] Result received from agent optimiser
[operator] Answering to chatbot: Desculpe-me, mas com os dados disponíveis atualmente, não foi possível gerar uma alocação otimizada.
[optimiser] Resultado da otimização salvo no banco de dados
[operator] Request received - Get Validation Result from Dialog
[operator] Chatbot of operator is requesting plan validation.
[assistant] Agent operator requesting validation result.
[validator] Agent assistant requesting validation result.
[assistant] Response received from agent validador
[operator] Answering to chatbot: Desculpe, não recebi o seu plano de alocação para validar. Por favor, envie novamente.
[assistant] Communication error -- no_applicable: Found a goal for which there is no applicable plan: +!saveResult(result("NULL"),"Desculpe, não recebi o seu plano de alocação para validar. Por favor, envie novamente.")[source(self)]
[operator] Request received - Get Validation Result from Dialog
[operator] Chatbot of operator is requesting plan validation.
[assistant] Agent operator requesting validation result.
[validator] Agent assistant requesting validation result.
[assistant] Response received from agent validador
[operator] Answering to chatbot: Desculpe, não recebi o seu plano de alocação para validar. Por favor, envie novamente.
[assistant] Communication error -- no_applicable: Found a goal for which there is no applicable plan: +!saveResult(result("NULL"),"Desculpe, não recebi o seu plano de alocação para validar. Por favor, envie novamente.")[source(self)]
[operator] Request received - Get Validation Result from Dialog
[operator] Chatbot of operator is requesting plan validation.
[assistant] Agent operator requesting validation result.
[validator] Agent assistant requesting validation result.
[assistant] Response received from agent validador
[operator] Answering to chatbot: O seu plano de alocação de leitos possui falhas. Houve um erro ao alocar os seguintes pacientes: Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo de Encaminhamento Agudo, e Especialidade Gastro como é o caso do paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira uma alocação otimizada?
[validator] Agent assistant requesting to save result in validation.
[database] Agent validator wants to update the validation result
[validator] Agent database is answering about the update.
[assistant] Retorno da validação salvo no banco de dados
[operator] Request received - Get Validation Result - confirm from Dialog
[operator] Chatbot of operator is requesting the confirmation of the allocation.
[assistant] Agent operator requesting allocation based on the last validation result.
[validator] Agent assistant requesting allocation.

[database] Agent validator wants to allocate patients using validation result
[validator] Agent database is answering about the allocation.
[operator] Answering to chatbot: Ok, pacientes alocados conforme solicitado

RV4JaCa log from specialist S2

{"msgId":"mid1","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"getSuggestionByPatient","prop1":{"arr":["\"Maria da Silva\""],"length":1},"isNegated":false}}
{"msgId":"mid2","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"getSuggestionByPatient","prop1":{"arr":["\"Maria da Silva\""],"length":1},"isNegated":false}}
{"msgId":"mid3","isReply":"nirt","performative":"assert","sender":"optimiser","receiver":"assistant","content":{"name":"suggestionByPatient","prop1":{"prop2":{"prop1":{"length":0},"name":"notAlloc","isNegated":false},"prop1":"\"true\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"arr":[{"prop2":"\"103a\"","prop1":"\"Maria da Silva\"","name":"alloc","isNegated":false}],"length":1},"name":"sugestedAllocation","isNegated":false}},"isNegated":false}}
{"msgId":"mid4","isReply":"nirt","performative":"question","sender":"optimiser","receiver":"database","content":{"name":"saveOptimiserResult","isNegated":false}}
{"msgId":"mid5","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Eu posso sugerir colocar o/a paciente Maria da Silva no leito 103a. Você gostaria que eu confirmasse essa alocação?\"","name":"answer","prop1":{"prop1":{"prop2":{"prop1":{"length":0},"name":"notAlloc","isNegated":false},"prop1":"\"true\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"arr":[{"prop2":"\"103a\"","prop1":"\"Maria da Silva\"","name":"alloc","isNegated":false}],"length":1},"name":"sugestedAllocation","isNegated":false}},"name":"suggestionByPatient","isNegated":false},"isNegated":false}}
{"msgId":"mid6","isReply":"nirt","performative":"assert","sender":"database","receiver":"optimiser","content":{"name":"savedOptimiserResult","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid7","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"cancelAllocByOptimization","isNegated":false}}
{"msgId":"mid8","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"cancelOpAlloc","isNegated":false}}
{"msgId":"mid9","isReply":"nirt","performative":"question","sender":"optimiser","receiver":"database","content":{"name":"cancelOpAlloc","isNegated":false}}
{"msgId":"mid10","isReply":"nirt","performative":"assert","sender":"database","receiver":"optimiser","content":{"name":"cancelOpAllocation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid11","isReply":"nirt","performative":"assert","sender":"optimiser","receiver":"assistant","content":{"name":"cancelOpAllocation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid12","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Ok, solicitação concluida sem alocar nenhum paciente.\"","name":"answer","prop1":{"prop1":"\"Success\"","name":"cancelOpAllocation","isNegated":false},"isNegated":false}}
{"msgId":"mid13","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"getSuggestionByPatient","prop1":{"arr":["\"Maria da Silva\""],"length":1},"isNegated":false}}
{"msgId":"mid14","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"getSuggestionByPatient","prop1":{"arr":["\"Maria da Silva\""],"length":1},"isNegated":false}}

{"msgId":"mid15","isReply":"nirt","performative":"assert","sender":"optim
iser","receiver":"assistant","content":{"name":"suggestionByPatient","pro
p1":{"prop2":{"prop1":{"length":0},"name":"notAlloc","isNegated":false},"
prop1":"\"true\"","name":"optimiserResult","isNegated":false,"prop3":{"pr
op1":{"arr":[{"prop2":"\"103a\"","prop1":"\"Maria da
Silva\"","name":"alloc","isNegated":false}],"length":1},"name":"sugestedA
llocation","isNegated":false}},"isNegated":false}}
{"msgId":"mid16","isReply":"nirt","performative":"question","sender":"opt
imiser","receiver":"database","content":{"name":"saveOptimiserResult","is
Negated":false}}
{"msgId":"mid17","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"Eu posso sugerir
colocar o/a paciente Maria da Silva no leito 103a. Você gostaria que eu
confirmasse essa
alocação?\"","name":"answer","prop1":{"prop1":{"prop2":{"prop1":{"length"
:0},"name":"notAlloc","isNegated":false},"prop1":"\"true\"","name":"optim
iserResult","isNegated":false,"prop3":{"prop1":{"arr":[{"prop2":"\"103a\"
","prop1":"\"Maria da
Silva\"","name":"alloc","isNegated":false}],"length":1},"name":"sugestedA
llocation","isNegated":false}},"name":"suggestionByPatient","isNegated":f
alse},"isNegated":false}}
{"msgId":"mid18","isReply":"nirt","performative":"assert","sender":"datab
ase","receiver":"optimiser","content":{"name":"savedOptimiserResult","pro
p1":"\"Success\"","isNegated":false}}
{"msgId":"mid19","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"name":"getOptimisedAllocation",
"isNegated":false}}
{"msgId":"mid20","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"optimiser","content":{"name":"getOptimisedAllocation"
,"isNegated":false}}
{"msgId":"mid21","isReply":"nirt","performative":"question","sender":"opt
imiser","receiver":"database","content":{"name":"saveOptimiserResult","is
Negated":false}}
{"msgId":"mid22","isReply":"nirt","performative":"assert","sender":"optim
iser","receiver":"assistant","content":{"prop2":{"prop1":{"arr":["\"Vitor
Almeida\"","\"Daniel
Souza\""],"length":2},"name":"notAlloc","isNegated":false},"name":"optimi
serResult","prop1":"\"false\"","isNegated":false,"prop3":{"prop1":{"arr":
[{"prop2":"\"205b\"","prop1":"\"Janaina de
Vargas\"","name":"alloc","isNegated":false},{"prop2":"\"205a\"","prop1":"
\"Maria da
Silva\"","name":"alloc","isNegated":false},{"prop2":"\"402a\"","prop1":"\
"Felipe
Pinto\"","name":"alloc","isNegated":false},{"prop2":"\"402b\"","prop1":"\
"Antônio
Marques\"","name":"alloc","isNegated":false},{"prop2":"\"301c\"","prop1":
"\"Vicente
Nunes\"","name":"alloc","isNegated":false},{"prop2":"\"103a\"","prop1":"\
"Marisa da
Costa\"","name":"alloc","isNegated":false}],"length":6},"name":"sugestedA
llocation","isNegated":false}}}
{"msgId":"mid23","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"Gerei uma alocação
otimizada, porém não conseguirei alocar todos os pacientes pois não
localizei leitos adequados para os pacientes Vitor Almeida, e Daniel
Souza. Você pode ver minha sugestão no menu \u0027Alocação
otimizada\u0027 aqui ao lado. Você quer que eu confirme essa
alocação?\"","name":"answer","prop1":{"prop2":{"prop1":{"arr":["\"Vitor

Almeida\"","\"Daniel
Souza\""],"length":2},"name":"notAlloc","isNegated":false},"prop1":"\"false\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"arr":[{"prop2":"\"205b\"","prop1":"\"Janaina de
Vargas\"","name":"alloc","isNegated":false},{"prop2":"\"205a\"","prop1":"\"Maria da
Silva\"","name":"alloc","isNegated":false},{"prop2":"\"402a\"","prop1":"\"Felipe
Pinto\"","name":"alloc","isNegated":false},{"prop2":"\"402b\"","prop1":"\"Antônio
Marques\"","name":"alloc","isNegated":false},{"prop2":"\"301c\"","prop1":"\"Vicente
Nunes\"","name":"alloc","isNegated":false},{"prop2":"\"103a\"","prop1":"\"Marisa da
Costa\"","name":"alloc","isNegated":false}],"length":6},"name":"sugestedAllocation","isNegated":false}},"isNegated":false}}
{"msgId":"mid24","isReply":"nirt","performative":"assert","sender":"database","receiver":"optimiser","content":{"name":"savedOptimiserResult","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid26","isReply":"nirt","performative":"question","sender":"assistant","receiver":"optimiser","content":{"name":"allocOpPatients","isNegated":false}}
{"msgId":"mid27","isReply":"nirt","performative":"question","sender":"optimiser","receiver":"database","content":{"name":"allocOpPatients","isNegated":false}}
{"msgId":"mid25","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"confirmAllocByOptimization","isNegated":false}}
{"msgId":"mid29","isReply":"nirt","performative":"assert","sender":"optimiser","receiver":"assistant","content":{"name":"allocatedOpPatients","prop1":"\"Erro\"","isNegated":false}}
{"msgId":"mid30","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"getValidationResult","isNegated":false}}
{"msgId":"mid31","isReply":"nirt","performative":"question","sender":"assistant","receiver":"validator","content":{"name":"getValidationResult","isNegated":false}}
{"msgId":"mid32","isReply":"nirt","performative":"assert","sender":"validator","receiver":"assistant","content":{"prop2":"\"false\"","name":"result","prop1":"\"c10473aa-426b-4eb3-b0bd-
ffd1c9cdee4a\"","isNegated":false,"prop3":{"arr":[{"prop2":"\"301b\"","prop1":"\"Vitor
Almeida\"","name":"err","isNegated":false,"prop3":{"arr":[{"prop2":"\"Tipo de Encaminhamento
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":"\"Agudo\""},{"prop2":"\"Especialidade
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":"\"Gastro\""}],"length":2}}],"length":1}}}
{"msgId":"mid33","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"O seu plano de alocação de leitos possui falhas. Houve um erro ao alocar os seguintes pacientes:
Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo de
Encaminhamento Agudo, e Especialidade Gastro como é o caso do
paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira uma alocação
otimizada?\"","name":"answer","prop1":{"prop2":"\"false\"","prop1":"\"c10473aa-426b-4eb3-b0bd-
ffd1c9cdee4a\"","name":"result","isNegated":false,"prop3":{"arr":[{"prop2

":"\"301b\"","prop1":"\"Vitor
Almeida\"","name":"err","isNegated":false,"prop3":{"arr":[{"prop2":"\"Tip
o de Encaminhamento
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":
"\"Agudo\""},{"prop2":"\"Especialidade
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":
"\"Gastro\""}],"length":2}}],"length":1}},"isNegated":false}}
{"msgId":"mid34","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"validator","content":{"prop2":"\"O seu plano de
alocação de leitos possui falhas. Houve um erro ao alocar os seguintes
pacientes: Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo
de Encaminhamento Agudo, e Especialidade Gastro como é o caso do
paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira
uma alocação otimizada?\"","name":"updateValidation","prop1":"\"c10473aa-
426b-4eb3-b0bd-ffd1c9cdee4a\"","isNegated":false}}
{"msgId":"mid35","isReply":"nirt","performative":"question","sender":"val
idator","receiver":"database","content":{"prop2":"\"O seu plano de
alocação de leitos possui falhas. Houve um erro ao alocar os seguintes
pacientes: Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo
de Encaminhamento Agudo, e Especialidade Gastro como é o caso do
paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira
uma alocação otimizada?\"","name":"updateValidation","prop1":"\"c10473aa-
426b-4eb3-b0bd-ffd1c9cdee4a\"","isNegated":false}}
{"msgId":"mid36","isReply":"nirt","performative":"assert","sender":"datab
ase","receiver":"validator","content":{"name":"updatedValidation","prop1"
:"\"Success\"","isNegated":false}}
{"msgId":"mid37","isReply":"nirt","performative":"assert","sender":"valid
ator","receiver":"assistant","content":{"name":"updatedValidation","prop1
":"\"Success\"","isNegated":false}}
{"msgId":"mid38","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"name":"getOptimisedAllocation",
"isNegated":false}}
{"msgId":"mid39","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"optimiser","content":{"name":"getOptimisedAllocation"
,"isNegated":false}}
{"msgId":"mid40","isReply":"nirt","performative":"question","sender":"opt
imiser","receiver":"database","content":{"name":"saveOptimiserResult","is
Negated":false}}
{"msgId":"mid41","isReply":"nirt","performative":"assert","sender":"optim
iser","receiver":"assistant","content":{"prop2":{"prop1":{"arr":["\"Vitor
Almeida\"","\"Daniel
Souza\""],"length":2},"name":"notAlloc","isNegated":false},"name":"optimi
serResult","prop1":"\"false\"","isNegated":false,"prop3":{"prop1":{"lengt
h":0},"name":"sugestedAllocation","isNegated":false}}}
{"msgId":"mid42","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"Desculpe-me, mas com os
dados disponíveis atualmente, não foi possível gerar uma alocação
otimizada.\"","name":"answer","prop1":{"prop2":{"prop1":{"arr":["\"Vitor
Almeida\"","\"Daniel
Souza\""],"length":2},"name":"notAlloc","isNegated":false},"prop1":"\"fal
se\"","name":"optimiserResult","isNegated":false,"prop3":{"prop1":{"lengt
h":0},"name":"sugestedAllocation","isNegated":false}},"isNegated":false}}
{"msgId":"mid43","isReply":"nirt","performative":"assert","sender":"datab
ase","receiver":"optimiser","content":{"name":"savedOptimiserResult","pro
p1":"\"Success\"","isNegated":false}}
{"msgId":"mid44","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"name":"getValidationResult","is
Negated":false}}

{"msgId":"mid45","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"validator","content":{"name":"getValidationResult","i
sNegated":false}}
{"msgId":"mid46","isReply":"nirt","performative":"assert","sender":"valid
ator","receiver":"assistant","content":{"name":"result","prop1":"\"NULL\"
","isNegated":false}}
{"msgId":"mid47","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"Desculpe, não recebi o
seu plano de alocação para validar. Por favor, envie
novamente.\"","name":"answer","prop1":{"prop1":"\"NULL\"","name":"result"
,"isNegated":false},"isNegated":false}}
{"msgId":"mid48","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"name":"getValidationResult","is
Negated":false}}
{"msgId":"mid49","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"validator","content":{"name":"getValidationResult","i
sNegated":false}}
{"msgId":"mid50","isReply":"nirt","performative":"assert","sender":"valid
ator","receiver":"assistant","content":{"name":"result","prop1":"\"NULL\"
","isNegated":false}}
{"msgId":"mid51","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"Desculpe, não recebi o
seu plano de alocação para validar. Por favor, envie
novamente.\"","name":"answer","prop1":{"prop1":"\"NULL\"","name":"result"
,"isNegated":false},"isNegated":false}}
{"msgId":"mid52","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"name":"getValidationResult","is
Negated":false}}
{"msgId":"mid53","isReply":"nirt","performative":"question","sender":"ass
istant","receiver":"validator","content":{"name":"getValidationResult","i
sNegated":false}}
{"msgId":"mid54","isReply":"nirt","performative":"assert","sender":"valid
ator","receiver":"assistant","content":{"prop2":"\"false\"","name":"resul
t","prop1":"\"3df9af07-55fb-4580-ba42-
e2a6b6f45e9d\"","isNegated":false,"prop3":{"arr":[{"prop2":"\"301b\"","pr
op1":"\"Vitor
Almeida\"","name":"err","isNegated":false,"prop3":{"arr":[{"prop2":"\"Tip
o de Encaminhamento
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":
"\"Agudo\""},{"prop2":"\"Especialidade
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":
"\"Gastro\""}],"length":2}}],"length":1}}}
{"msgId":"mid55","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"O seu plano de alocação
de leitos possui falhas. Houve um erro ao alocar os seguintes pacientes:
Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo de
Encaminhamento Agudo, e Especialidade Gastro como é o caso do
paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira
uma alocação
otimizada?\"","name":"answer","prop1":{"prop2":"\"false\"","prop1":"\"3df
9af07-55fb-4580-ba42-
e2a6b6f45e9d\"","name":"result","isNegated":false,"prop3":{"arr":[{"prop2
":"\"301b\"","prop1":"\"Vitor
Almeida\"","name":"err","isNegated":false,"prop3":{"arr":[{"prop2":"\"Tip
o de Encaminhamento
\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":
"\"Agudo\""},{"prop2":"\"Especialidade

\"","prop1":"\"missingPositive\"","name":"mot","isNegated":false,"prop3":"\"Gastro\""}],"length":2}}],"length":1}},"isNegated":false}}
{"msgId":"mid56","isReply":"nirt","performative":"question","sender":"assistant","receiver":"validator","content":{"prop2":"\"O seu plano de alocação de leitos possui falhas. Houve um erro ao alocar os seguintes pacientes: Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo de Encaminhamento Agudo, e Especialidade Gastro como é o caso do paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira uma alocação otimizada?\"","name":"updateValidation","prop1":"\"3df9af07-55fb-4580-ba42-e2a6b6f45e9d\"","isNegated":false}}
{"msgId":"mid57","isReply":"nirt","performative":"question","sender":"validator","receiver":"database","content":{"prop2":"\"O seu plano de alocação de leitos possui falhas. Houve um erro ao alocar os seguintes pacientes: Vitor Almeida no leito 301b - Pois o leito 301b não é de Tipo de Encaminhamento Agudo, e Especialidade Gastro como é o caso do paciente. Devo confirmar a alocação mesmo assim ou prefere que eu sugira uma alocação otimizada?\"","name":"updateValidation","prop1":"\"3df9af07-55fb-4580-ba42-e2a6b6f45e9d\"","isNegated":false}}
{"msgId":"mid58","isReply":"nirt","performative":"assert","sender":"database","receiver":"validator","content":{"name":"updatedValidation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid59","isReply":"nirt","performative":"assert","sender":"validator","receiver":"assistant","content":{"name":"updatedValidation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid60","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"allocValPatients","isNegated":false}}
{"msgId":"mid61","isReply":"nirt","performative":"question","sender":"assistant","receiver":"validator","content":{"name":"allocByValidation","isNegated":false}}
{"msgId":"mid62","isReply":"nirt","performative":"question","sender":"validator","receiver":"database","content":{"name":"allocByValidation","isNegated":false}}
{"msgId":"mid63","isReply":"nirt","performative":"assert","sender":"database","receiver":"validator","content":{"name":"allocByValidation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid64","isReply":"nirt","performative":"assert","sender":"validator","receiver":"assistant","content":{"name":"allocByValidation","prop1":"\"Success\"","isNegated":false}}
{"msgId":"mid65","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Ok, pacientes alocados conforme
solicitado\"","name":"answer","prop1":{"prop1":"\"Success\"","name":"allocByValidation","isNegated":false},"isNegated":false}}

Appendix C

Evaluation of Expressivity of the Framework – Logs

MAS log

```
[NetworkListener] Started listener bound to [{0}]
[HttpServer] [{0}] Started.
[Cartago] Workspace wp created.
[Cartago] getJoinedWorkspaces: [main]
[Cartago] artifact dial4jaca:
br.pucrs.smart.Dial4JaCa.Dial4JaCaArtifact() at wp created.
[Cartago] artifact rv4jaca: br.pucrs.smart.RV4JaCa.RV4JaCaArtifact() at
wp created.
[Onto4JaCaArtifact] Importing ontology from src/resources/hsl-
ontology.owl
[Onto4JaCaArtifact] Ontology ready!
[Cartago] artifact onto4jaca:
br.pucrs.smart.Onto4JaCa.mas.Onto4JaCaArtifact("src/resources/hsl-
ontology.owl") at wp created.
[Cartago] artifact pddl: br.pucrs.smart.validator.ValidatorArtifact() at
wp created.
[Cartago] artifact postgres: br.pucrs.smart.postgresql.PostgresArtifact()
at wp created.
[ontology_specialist] joinned workspace wp
[ontology_specialist] focusing on artifact onto4jaca (at workspace wp)
using namespace default
[ontology_specialist] focusing on artifact postgres (at workspace wp)
using namespace default
[database] joinned workspace wp
[database] focusing on artifact postgres (at workspace wp) using
namespace default
[assistant] Assistant agent enabled.
[validator] joinned workspace wp
[optimiser] Optimiser agent enabled.
[operator] joinned workspace wp
[operator] focusing on artifact dial4jaca (at workspace wp) using
namespace default
[validator] focusing on artifact pddl (at workspace wp) using namespace
default
[validator] Validator agent enabled.
[operator] Communication specialist agent enabled.
[nurse] joinned workspace wp
[database] Database specialist agent enabled.
[nurse] focusing on artifact dial4jaca (at workspace wp) using namespace
default
[nurse] Communication specialist agent enabled.
[ontology_specialist] Iniciando busca no banco de dados
[ontology_specialist] Busca no banco de dados finalizada
[ontology_specialist] Agent ontology_specialist enabled.
[nurse] Request received - Default Welcome Intent from Dialog
[nurse] Params: []
[operator] Request received - Default Welcome Intent from Dialog
[operator] Params: []
[operator] Request received - Default Welcome Intent from Dialog
[operator] Params: []
[operator] Request received - Default Welcome Intent from Dialog
[operator] Params: []
[nurse] Request received - Default Welcome Intent from Dialog
[nurse] Params: []
[operator] Request received - Default Welcome Intent from Dialog
[operator] Params: []
```

```
[operator] Request received - Allocate Patient from Dialog
[operator] Params: [param("paciente","2044429"),param("leito","3132")]
[operator] Chatbot of operator is requesting an allocation.
[assistant] Agent operator asks to allocate patient 2044429 to bed 3132
[operator] Answering to chatbot: Antes eu preciso verificar se esse leito
é adequado para esse paciente, pode ser?
[operator] Request received - Allocate Patient - verify suitability from
Dialog
[operator] Params: []
[operator] Chatbot of operator is requesting to verify suitability:
paciente: 2044429, leito: 3132
[ontology_specialist] Verifying if adequado("3132","2044429").
[ontology_specialist] Verifying if inadequado("3132","2044429").
[operator] Answering to chatbot: Não, esse leito não é adequado.
[operator] Request received - Allocate Patient - verify suitability - why
from Dialog
[operator] Params: []
[operator] Chatbot of operator is requesting explanation about
suitability of 3132 leito to 2044429
[ontology_specialist] Received explain request for explaning the
predicate: inadequado("3132","2044429")
[ontology_specialist] !getAnswer
[ontology_specialist] Explanation for adequado("3132","2044429") not
found.
[ontology_specialist] Found argument for: inadequado("3132","2044429")
[operator] Answering to chatbot: O leito 3132 não é adequado para o(a)
paciente 2044429 pois a cobertura do(a) paciente é ENFERMARIA e esse
leito é UTI ADULTO GERAL.
[operator] Request received - Allocate Patient - verify suitability - why
- allocate anyway from Dialog
[operator] Params: []
[operator] Chatbot of operator is requesting to allocate anyway.
[assistant] Agent operator asks to allocate a patient to a bed anyway
[operator] Answering to chatbot: Desculpe, você não tem autorização para
efetuar essa alocação, mas você pode pedir para a gestora abrir uma
exceção
[nurse] Request received - Default Welcome Intent from Dialog
[nurse] Params: []
[nurse] Request received - Register Exception from Dialog
[nurse] Params: [param("paciente","2044429"),param("leito","3132")]
[nurse] Chatbot of nurse is requesting to register an exception:
paciente: 2044429, leito: 3132
[assistant] Agent nurse asks to register exception on patient 2044429 and
bed 3132
[ontology_specialist] Registering exception for bed 3132 and patient
2044429
[assistant] Agent ontology_specialist informs that the exception on
patient 2044429 and bed 3132 was registered.
[nurse] Answering to chatbot: Ok, estou registrando a exceção solicitada
[operator] Request received - Verify Suitability from Dialog
[operator] Params: [param("paciente","2044429"),param("leito","3132")]
[operator] Chatbot of operator is requesting to verify suitability:
paciente: 2044429, leito: 3132
[ontology_specialist] Verifying if adequado("3132","2044429").
[operator] Answering to chatbot: Sim, esse leito é adequado.
[operator] Request received - Verify Suitability - why from Dialog
[operator] Params: []
```

```
[operator] Chatbot of operator is requesting explanation about
suitability of 3132 leito to 2044429
[ontology_specialist] Received explain request for explaning the
predicate: inadequado("3132","2044429")
[ontology_specialist] !getAnswer
[ontology_specialist] Found argument for: adequado("3132","2044429")
[operator] Answering to chatbot: O leito 3132 é adequado para o(a)
paciente 2044429 pois o(a) gestor(a) abriu uma exceção para esse caso.
[operator] Request received - getSuperlativeInfo1 from Dialog
[operator] Params: []
[operator] Chatbot of operator is asking about the smallest room with
fewer patients allocated.
[assistant] Agent operator asks about the smallest room with fewest
occupants.
[database] Agent assistant wants get the smallest room with fewest
occupants.
[assistant] Agent database answer about the smallest room with fewest
occupants.
[operator] Answering to chatbot: O menor quarto com o menor número de
ocupantes é o 118J
[operator] Request received - getSuperlativeInfo3 from Dialog
[operator] Params: []
[operator] Chatbot of operator is asking about the smallest room with
most patients allocated.
[assistant] Agent operator asks about the biggest room with most
occupants.
[database] Agent assistant wants get the biggest room with most
occupants.
[assistant] Agent database answer about the biggest room with most
occupants.
[operator] Answering to chatbot: O maior quarto com o maior número de
ocupantes é o 759
[operator] Request received - Get Information About Beds from Dialog
[operator] Params:
[param("status_leito","Vago"),param("genero","FEMININO"),param("tipo_espe
cialidade",""),param("acomodacao",""),param("faixa_etaria","")]
[operator] Chatbot of operator is asking for information about beds.
[assistant] Agent operator asks about a bed
leito_e_do_genero(Bed,"FEMININO")leito_e_da_faixa_etaria(Bed,"")leito_e_d
o_tipo_especialidade(Bed,"")e_de_acomodacao(Bed,"")possui_status(Bed,"Vag
o")
[database] Agent assistant wants a bed.
[assistant] Agent database answered about the bed.
[operator] Answering to chatbot: Localizei o leito 3132 que se encaixa na
sua solicitação. Você gostaria de alocar um paciente nele?
[operator] Request received - Default Welcome Intent from Dialog
[operator] Params: []
```

RV4JaCa log

{"msgId":"mid1","isReply":"nirt","performative":"question","sender":"oper
ator","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"alo
car","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid2","isReply":"nirt","performative":"assert","sender":"assist
ant","receiver":"operator","content":{"prop2":"\"Antes eu preciso
verificar se esse leito é adequado para esse paciente, pode
ser?\"","name":"answer","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\""
,"name":"verifySuitability","isNegated":false},"isNegated":false}}
{"msgId":"mid3","isReply":"nirt","performative":"question","sender":"oper
ator","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"ade
quado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid4","isReply":"nirt","performative":"question","sender":"assi
stant","receiver":"ontology_specialist","content":{"prop2":"\"2044429\"",
"name":"adequado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid6","isReply":"nirt","performative":"assert","sender":"assist
ant","receiver":"operator","content":{"prop2":"\"Não, esse leito não é
adequado.\"","name":"answer","prop1":{"prop2":"\"2044429\"","prop1":"\"31
32\"","name":"adequado","isNegated":false},"isNegated":false}}
{"msgId":"mid5","isReply":"nirt","performative":"assert","sender":"ontolo
gy_specialist","receiver":"assistant","content":{"prop2":"\"2044429\"","n
ame":"adequado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid7","isReply":"nirt","performative":"question","sender":"oper
ator","receiver":"assistant","content":{"name":"explain","prop1":{"prop2"
:"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated":false},
"isNegated":false}}
{"msgId":"mid8","isReply":"nirt","performative":"question","sender":"assi
stant","receiver":"ontology_specialist","content":{"name":"explain","prop
1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegat
ed":false},"isNegated":false}}
{"msgId":"mid10","isReply":"nirt","performative":"assert","sender":"assis
tant","receiver":"operator","content":{"prop2":"\"O leito 3132 não é
adequado para o(a) paciente 2044429 pois a cobertura do(a) paciente é
ENFERMARIA e esse leito é UTI ADULTO GERAL.
\"","name":"answer","prop1":{"prop2":{"arr":[{"prop2":{"arr":[{"prop1":"\
"2044429\"","name":"paciente","isNegated":false},{"prop1":"\"3132\"","nam
e":"leito","isNegated":false},{"prop2":"\"ENFERMARIA\"","prop1":"\"204442
9\"","name":"possui_cobertura","isNegated":false},{"prop2":"\"UTI ADULTO
GERAL\"","prop1":"\"3132\"","name":"e_de_acomodacao","isNegated":false},{
"prop2":"\"UTI ADULTO
GERAL\"","prop1":"\"ENFERMARIA\"","name":"differentFrom","isNegated":fals
e},{"prop2":"\"NONE\"","prop1":"\"UTI ADULTO
GERAL\"","name":"differentFrom","isNegated":false},{"prop2":"\"NONE\"","p
rop1":"\"ENFERMARIA\"","name":"differentFrom","isNegated":false}],"length
":7},"prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado
","isNegated":false},"name":"defeasible_rule","isNegated":false}],"length
":1},"prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado
","isNegated":false},"name":"explaining","isNegated":false},"isNegated":f
alse}}
{"msgId":"mid9","isReply":"nirt","performative":"assert","sender":"ontolo
gy_specialist","receiver":"assistant","content":{"name":"explain","prop1"
:{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated
":false},"isNegated":false}}
{"msgId":"mid11","isReply":"nirt","performative":"question","sender":"ope
rator","receiver":"assistant","content":{"prop2":"Patient","name":"alocar
","prop1":"B","isNegated":false}}

{"msgId":"mid12","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Desculpe, você não tem autorização para efetuar essa alocação, mas você pode pedir para a gestora abrir uma
exceção\"","name":"answer","prop1":{"prop1":{"prop2":"Patient","prop1":"B","name":"alocar","isNegated":false},"name":"askPermission","isNegated":false},"isNegated":false}}
{"msgId":"mid13","isReply":"nirt","performative":"question","sender":"nurse","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"exception","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid14","isReply":"nirt","performative":"question","sender":"assistant","receiver":"ontology_specialist","content":{"prop2":"\"2044429\"","name":"nurse_exception","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid15","isReply":"nirt","performative":"assert","sender":"ontology_specialist","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"nurse_exception","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid16","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"nurse","content":{"prop2":"\"Ok, estou registrando a exceção
solicitada\"","name":"answer","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"nurse_exception","isNegated":false},"isNegated":false}}
{"msgId":"mid17","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"adequado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid18","isReply":"nirt","performative":"question","sender":"assistant","receiver":"ontology_specialist","content":{"prop2":"\"2044429\"","name":"adequado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid19","isReply":"nirt","performative":"assert","sender":"ontology_specialist","receiver":"assistant","content":{"prop2":"\"2044429\"","name":"adequado","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid20","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Sim, esse leito é adequado.\"","name":"answer","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"adequado","isNegated":false},"isNegated":false}}
{"msgId":"mid21","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"explain","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated":false},"isNegated":false}}
{"msgId":"mid22","isReply":"nirt","performative":"question","sender":"assistant","receiver":"ontology_specialist","content":{"name":"explain","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated":false},"isNegated":false}}
{"msgId":"mid23","isReply":"nirt","performative":"assert","sender":"ontology_specialist","receiver":"assistant","content":{"name":"explain","prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated":false},"isNegated":false}}
{"msgId":"mid24","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"O leito 3132 é adequado para o(a) paciente 2044429 pois o(a) gestor(a) abriu uma exceção para esse caso.
\"","name":"answer","prop1":{"prop2":{"arr":[{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"adequado","isNegated":false}],"length":1},"prop1":{"prop2":"\"2044429\"","prop1":"\"3132\"","name":"inadequado","isNegated":false},"name":"explaining","isNegated":false},"isNegated":false}}
{"msgId":"mid25","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"smallestRoom","isNegated":false}}

{"msgId":"mid26","isReply":"nirt","performative":"question","sender":"assistant","receiver":"database","content":{"name":"smallestRoom","isNegated":false}}
{"msgId":"mid27","isReply":"nirt","performative":"assert","sender":"database","receiver":"assistant","content":{"name":"smallestRoom","prop1":"\"118J\"","isNegated":false}}
{"msgId":"mid28","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"O menor quarto com o menor número de ocupantes é o 118J\"","name":"answer","prop1":{"prop1":"\"118J\"","name":"smallestRoom","isNegated":false},"isNegated":false}}
{"msgId":"mid29","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"biggestRoom","isNegated":false}}
{"msgId":"mid30","isReply":"nirt","performative":"question","sender":"assistant","receiver":"database","content":{"name":"biggestRoom","isNegated":false}}
{"msgId":"mid31","isReply":"nirt","performative":"assert","sender":"database","receiver":"assistant","content":{"name":"biggestRoom","prop1":"\"759\"","isNegated":false}}
{"msgId":"mid32","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"O maior quarto com o maior número de ocupantes é o 759\"","name":"answer","prop1":{"prop1":"\"759\"","name":"biggestRoom","isNegated":false},"isNegated":false}}
{"msgId":"mid33","isReply":"nirt","performative":"question","sender":"operator","receiver":"assistant","content":{"name":"getBed","isNegated":false}}
{"msgId":"mid34","isReply":"nirt","performative":"question","sender":"assistant","receiver":"database","content":{"name":"getBed","isNegated":false}}
{"msgId":"mid35","isReply":"nirt","performative":"assert","sender":"database","receiver":"assistant","content":{"name":"getBed","prop1":"\"3132\"","isNegated":false}}
{"msgId":"mid36","isReply":"nirt","performative":"assert","sender":"assistant","receiver":"operator","content":{"prop2":"\"Localizei o leito 3132 que se encaixa na sua solicitação. Você gostaria de alocar um paciente nele?\"","name":"answer","prop1":{"prop1":"\"3132\"","name":"getBed","isNegated":false},"isNegated":false}}

Bibliography

[AB18] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[ABdR+20] Zeynep Akata, Dan Balliet, Maarten de Rijke, Frank Dignum, Virginia Dignum, Guszti Eiben, Antske Fokkens, Davide Grossi, Koen Hindriks, Holger Hoos, 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.

[ABS13] Suket Arora, Kamaljeet Batra, and Sarabjit Singh. Dialogue system: A brief review. *arXiv preprint arXiv:1306.4134*, abs/1306.4134, 2013.

[AFFM21] Davide Ancona, Luca Franceschini, Angelo Ferrando, and Viviana Mascardi. RML: Theory and Practice of a Domain Specific Language for Runtime Verification. *Science of Computer Programming*, 205:102610, 2021.

[AFM16] Davide Ancona, Angelo Ferrando, and Viviana Mascardi. Comparing trace expressions and linear temporal logic for runtime verification. In Erika Ábrahám, Marcello M. Bonsangue, and Einar Broch Johnsen, editors, *Theory and Practice of Formal Methods - Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday*, volume 9660 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2016.

[AFM17] Davide Ancona, Angelo Ferrando, and Viviana Mascardi. Parametric runtime verification of multiagent systems. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1457–1459. ACM, 2017.

[AH19] Cleber Jorge Amaral and Jomi Fred Hübner. Jacamo-web is on the fly: An interactive multi-agent system ide. *Proceedings of the EMAS*, 2019.

[AHK20] Cleber Jorge Amaral, Jomi Fred Hübner, and Timotheus Kampik. Towards jacamo-rest: A resource-oriented abstraction for managing multi-agent systems, 2020.

[AJSS17] Amir Ahmadi-Javid, Pardis Seyedi, and Siddhartha S Syam. A survey of healthcare facility location. *Computers & Operations Research*, 79:223–263, 2017.

[AMHB04] Davide Ancona, Viviana Mascardi, Jomi Fred Hübner, and Rafael H. Bordini. Coo-agentspeak: Cooperation in agentspeak through plan exchange. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems, New York, USA*, pages 696–705. IEEE Computer Society, 2004.

[AMP00] Leila Amgoud, Nicolas Maudet, and Simon Parsons. Modeling dialogues using argumentation. In *ICMAS*, pages 31–38. IEEE Computer Society, 2000.

[ANCF19] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[AW03] Grigoris Antoniou and Gerd Wagner. Rules and defeasible reasoning on the semantic web. In *Rules and Rule Markup Languages for the Semantic Web, Second International Workshop, RuleML 2003, Sanibel Island, FL, USA, October 20, 2003, Proceedings*, pages 111–120, 2003.

[BA11] Elizabeth Black and Katie Atkinson. Choosing persuasive arguments for action. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 905–912. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[BBH+13] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6):747–761, October 2013.

[BBHR20] Olivier Boissier, Rafael H Bordini, Jomi Hubner, and Alessandro Ricci. *Multi-agent oriented programming: programming multi-agent systems using JaCaMo*. MIT Press, 2020.

[BCD07] T. J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artif. Intell.*, 171(10-15):619–641, July 2007.

[BCH17] Elizabeth Black, Amanda Jane Coles, and Christopher Hampson. Planning for persuasion. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 933–942, 2017.

[Ber14] Tudor Berariu. An argumentation framework for bdi agents. In *Intelligent Distributed Computing VII*, pages 343–354. Springer, 2014.

[BFFR18] Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In Ezio Bartocci and Yliès Falcone, editors, *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2018.

[BFVW06] Rafael H. Bordini, Michael Fisher, Willem Visser, and Michael J. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[BH07] Elizabeth Black and Anthony Hunter. A generative inquiry dialogue system. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 241. ACM, 2007.

[BH08] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. The MIT Press, 2008.

[BHB17] Gregor Behnke, Daniel Höller, and Susanne Biundo. This is a solution! (... But is it though?) - verifying solutions of hierarchical planning problems. In *27th International Conference on Automated Planning and Scheduling*, pages 20–28, 2017.

[BHHK03] M. Becker, C. Heine, Rainer Herrler, and K.-H. Krempels. *OntHoS - an Ontology for Hospital Scenarios.*, chapter 7, pages 87–103. Birkhäuser Verlag, 2003. In: Moreno, A.; Nealon, J. (Eds.): Applications of Software Agent Technology in the Health Care Domain. Whitestein Series in Software Agent Technologies (WSSAT).

[BHW07a] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, October 2007.

[BHW07b] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[BJG+17] Jayalakshmi Baskar, Rebecka Janols, Esteban Guerrero, Juan Carlos Nieves, and Helena Lindgren. A multipurpose goal model for personalised digital coaching. In *Agents and Multi-Agent Systems for Health Care*, pages 94–116. Springer, 2017.

[BLS07] Andreas Bauer, Martin Leucker, and Christian Schallhart. The good, the bad, and the ugly, but how ugly is ugly? In Oleg Sokolsky and Serdar Taşıran, editors, *Runtime Verification*, pages 126–138, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BMT+22] Alessandro Bigoni, Ana Maria Malik, Renato Tasca, Mariana Baleeiro Martins Carrera, Laura Maria Cesar Schiesari, Dante Dianezi Gambardella, and Adriano Massuda. Brazil's health system functionality amidst of the covid-19 pandemic: An analysis of resilience. *The Lancet Regional Health-Americas*, 10:100222, 2022.

[BN03] Franz Baader and Werner Nutt. Basic description logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

[BTV20] Pietro Baroni, Francesca Toni, and Bart Verheij. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games: 25 years later. *Argument & Computation*, (Preprint):1–14, 2020.

[Cla97] Edmund M Clarke. Model checking. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 54–56. Springer, 1997.

[CLYT17] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[Coh19] Philip Cohen. Foundations of collaborative task-oriented dialogue: What's in a slot? In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 198–209, Stockholm, Sweden, September 2019. Association for Computational Linguistics.

[CP79] Philip R. Cohen and C. Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[CQTF17] Ching-Ying Cheng, Xiaobei Qian, Shih-Huan Tseng, and Li-Chen Fu. Recommendation dialogue system through pragmatic argumentation. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 335–340. IEEE, 2017.

[CR20] Angelo Croatti and Alessandro Ricci. Mobile apps as personal assistant agents: the jaca-android framework for programming agents-based applications on mobile devices. *Autonomous Agents and Multi-Agent Systems*, 34(2):1–27, 2020.

[Cus22] Marcelo da S Custódio. Rasa4jaca: Uma interface entre sistemas multiagentes e tecnologias chatbots open source, 2022.

[DB17] Frank Dignum and Floris Bex. Creating dialogues using argumentation and social practices. In *International Conference on Internet Science*, pages 223–235. Springer, 2017.

[DDE20] Ivan Donadello, Mauro Dragoni, and Claudio Eccher. Explaining reasoning algorithms with persuasiveness: a case study for a behavioural change system. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 646–653, 2020.

[dOGPB+20] Vágner de Oliveira Gabriel, Alison R. Panisson, Rafael H. Bordini, Diana Francisca Adamatti, and Cléo Zanella Billa. Reasoning in BDI agents using toulmin's argumentation model. *Theor. Comput. Sci.*, 805:76–91, 2020.

[dS17] Ministério da Saúde. *Manual de implantação e implementação : núcleo interno de regulação para Hospitais Gerais e Especializados [recurso eletrônico] / Ministério da Saúde, Secretaria de Atenção à Saúde, Departamento de Atenção Hospitalar e de Urgência*. Ministério da Saúde, Brasília, 2017. <http://portalarquivos2.saude.gov.br/images/PDF/2018/marco/28/Manual-NIR—Versao-digital-RGB.PDF>.

[dSCVMB21] Mateus da Silveira Colissi, Renata Vieira, Viviana Mascardi, and Rafael H. Bordini. A chatbot that uses a multi-agent organization to support collaborative learning. In Constantine Stephanidis, Margherita Antona, and Stavroula Ntoa, editors, *HCI International 2021 - Posters*, pages 31–38, Cham, 2021. Springer International Publishing.

[Dun95a] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[Dun95b] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[DWV11] Harmen De Weerd and Bart Verheij. The advantage of higher-order theory of mind in the game of limited bidding. In *Proceedings of the Workshop on Reasoning about Other Minds*, volume 751, pages 149–164, 2011.

[dWVV12] Harmen de Weerd, Rineke Verbrugge, and Bart Verheij. Higher-order social cognition in rock-paper-scissors: A simulation study. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 1195–1196, 2012.

[ECG+19] Debora Engelmann, Julia Couto, Vagner Gabriel, Renata Vieira, and Rafael Bordini. Towards an ontology to support decision-making in hospital bed allocation. In *Proceedings of 31st International Conference on Software Engineering & Knowledge Engineering*, pages 71–74, 2019.

[ECPB21] Débora C Engelmann, Lucca Dornelles Cezar, Alison R Panisson, and Rafael H Bordini. A conversational agent to support hospital bed allocation. In *Brazilian Conference on Intelligent Systems*, pages 3–17. Springer, 2021.

[EDK+21a] Débora Engelmann, Juliana Damasio, Tabajara Krausburg, Olimar Borges, Lucca Dornelles Cezar, Alison R Panisson, and Rafael H Bordini. Dial4jaca–a demonstration. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 346–350. Springer, 2021.

[EDK+21b] Débora Engelmann, Juliana Damasio, Tabajara Krausburg, Olimar Borges, Mateus Colissi, Alison R Panisson, and Rafael H Bordini. Dial4jaca–a communication interface between multi-agent systems and chatbots. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 77–88. Springer, 2021.

[EDP+22] Débora Engelmann, Juliana Damasio, Alison R Panisson, Viviana Mascardi, and Rafael H Bordini. Argumentation as a method for explainable ai: A systematic literature review. In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE, 2022.

[EFP+22] Debora C Engelmann, Angelo Ferrando, Alison R Panisson, Davide Ancona, Rafael H Bordini, and Viviana Mascardi. Rv4jaca – runtime verification for multi-agent systems. In *Proceedings of the Second Workshop on Agents and Robots for reliable Engineered Autonomy (AREA-2022), Austria*, 2022.

[Eng19] Débora Cristina Engelmann. *An interactive agent to support hospital bed allocation based on plan validation*. dissertation, Pontifícia Universidade Católica do Rio Grande do Sul, 2019.

[Eng20] Debora C. Engelmann. Conversational agents based on argumentation theory and ontologies. In *Proceedings of the Fourth Summer School on Argumentation: Computational and Linguistic Perspectives (SSA 2020)*, pages 10–12, 2020.

[eOdVAP20] BRP e Oliveira, JA de Vasconcelos, JFF Almeida, and LR Pinto. A simulation-optimisation approach for hospital beds allocation. *International Journal of Medical Informatics*, 141:104174, 2020.

[EPV+] Débora C Engelmann, Alison R Panisson, Renata Vieira, Jomi F Hübner, Viviana Mascardi, and Rafael H Bordini. Maids — a framework for the development of multi-agent intentional dialogue systems. (unpublished - under review for AAMAS 2023).

[ERS+18] Kai Essers, Rebecca Rogers, Jackie Sturt, Elizabeth I Sklar, and Elizabeth Black. Assessing the posture prototype: A late-breaking report on patient views. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, pages 344–346, 2018.

[Esf23] Zeinab Namakizadeh Esfahani. *Onto2Conv: An application for boosting cognitive conversational agents*. Master thesis (unpublished), University of Genoa, 2023.

[EW22] Amir Elalouf and Guy Wachtel. Queueing problems in emergency departments: A review of practical approaches and research methodologies. In *Operations Research Forum*, volume 3, pages 1–46. Springer, 2022.

[FGM03] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From agents to organizations: An organizational view of multi-agent systems. In Paolo Giorgini, Jörg P. Müller, and James Odell, editors, *AOSE*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer, 2003.

[FPE+22] Carlos Eduardo A Ferreira, Alison R Panisson, Débora C Engelmann, Renata Vieira, Viviana Mascardi, and Rafael H Bordini. Explaining semantic reasoning using argumentation. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 153–165. Springer, 2022.

[FPH+15] A. Freitas, A. R. Panisson, L. Hilgert, F. Meneguzzi, R. Vieira, and R. H. Bordini. Integrating ontologies with multi-agent systems through cartago artifacts. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 143–150, Dec 2015.

[FPH+17] Artur Freitas, Alison R Panisson, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Applying ontologies to the development and execution of multi-agent systems. In *Web Intelligence*, volume 15(4), pages 291–302. IOS Press, 2017.

[FRS18] Amal El Fallah-Seghrouchni, Alessandro Ricci, and Tran Cao Son, editors. *Engineering Multi-Agent Systems - 5th International Workshop, EMAS 2017, Sao Paulo, Brazil, May 8-9, 2017, Revised Selected Papers*, volume 10738 of *Lecture Notes in Computer Science*. Springer, 2018.

[G+12] Alvin I Goldman et al. Theory of mind. *The Oxford handbook of philosophy of cognitive science*, 1:1–25, 2012.

[GdCR+18a] M. da S. Grübler, C. A. da Costa, R. Righi, S. Rigo, and L. Chiwiacowsky. A hospital bed allocation hybrid model based on situation awareness. *CIN: Computers, Informatics, Nursing*, 36(5):249–255, May 2018.

[GdCR+18b] M. da S. Grübler, C. A. da Costa, R. Righi, S. Rigo, and L. Chiwiacowsky. A hospital bed allocation hybrid model based on situation awareness. *Computers, Informatics, Nursing*, 36:249–255, May 2018.

[GG19] Laura Giordano and Valentina Gliozzi. Strengthening the rational closure for description logics: An overview. In Alberto Casagrande and Eugenio G. Omodeo, editors, *Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019*, volume 2396 of *CEUR Workshop Proceedings*, pages 68–81. CEUR-WS.org, 2019.

[GM22] Andrea Gatti and Viviana Mascardi. Towards VEsNA, a framework for managing virtual environments via natural language agents. *Electronic Proceedings in Theoretical Computer Science*, 362:65–80, jul 2022.

[GMAB04] Guido Governatori, Michael J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logic. *J. Log. Comput.*, 14(5):675–702, 2004.

[GMBE+13] Adela Grando, Laura Moss, Gemma Bel-Enguix, M Dolores Jiménez-López, and John Kinsella. Argumentation-based dialogue systems for medical training. In *Where Humans Meet Machines*, pages 213–232. Springer, 2013.

[GMF+03] John H Gennari, Mark A Musen, Ray W Fergerson, William E Grosso, Monica Crubézy, Henrik Eriksson, Natalya F Noy, and Samson W Tu. The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123, 2003.

[Gru95] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

[GSC+19a] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science Robotics*, 4(37), 2019.

[GSC+19b] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120, 2019.

[Gun17] David Gunning. Explainable artificial intelligence (xai). *Defense advanced research projects agency (DARPA), nd Web*, 2(2):1, 2017.

[HDM+12] Nabila Hadidi, Yannis Dimopoulos, Pavlos Moraitis, et al. Tactics and concessions for argumentation-based negotiation. In *COMMA*, pages 285–296, 2012.

[HJ02] C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, Feb 2002.

[HL04] Bryan Horling and Victor R. Lesser. A survey of multi-agent organizational paradigms. *Knowledge Eng. Review*, 19(4):281–316, 2004.

[HLF04] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *ICTAI*, pages 294–301. IEEE Computer Society, 2004.

[HPSB+04] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31, 2004.

[HSB07] Jomi F Hubner, Jaime S Sichman, and Olivier Boissier. Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3-4):370–395, 2007.

[HSM+13] Christos Hadjinikolis, Yiannis Siantos, Sanjay Modgil, Elizabeth Black, and Peter McBurney. Opponent modelling in persuasion dialogues. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 164–170, 2013.

[HTFJ19] Joana Hois, Dimitra Theofanou-Fuelbier, and Alischa Janine Junk. How to achieve explainability and transparency in human ai interaction. In *International Conference on Human-Computer Interaction*, pages 177–183. Springer, 2019.

[KB08a] Thomas Klapiscak and Rafael H Bordini. Jasdl: A practical programming approach combining agent and semantic web technologies. In *International Workshop on Declarative Agent Languages and Technologies*, pages 91–110. Springer, 2008.

[KB08b] Thomas Klapiscak and Rafael H. Bordini. JASDL: A practical programming approach combining agent and semantic web technologies. In *Declarative Agent Languages and Technologies VI, 6th International Workshop, DALT 2008, Estoril, Portugal, May 12, 2008, Revised Selected and Invited Papers*, pages 91–110, 2008.

[KC04] Sanjeev Kumar and Philip R. Cohen. STAPLE: an agent programming language based on the joint intention theory. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 August 2004, New York, NY, USA*, pages 1390–1391, 2004.

[KCB+19] Nadin Kökciyan, Martin Chapman, Panagiotis Balatsoukas, Isabel Sassoon, Kai Essers, Mark Ashworth, Vasa Curcin, Sanjay Modgil, Simon Parsons, and Elizabeth I Sklar. A collaborative decision support tool for managing chronic conditions. In *MedInfo*, pages 644–648, 2019.

[KCD+18] Tabajara Krausburg, Rafael Cauê Cardoso, Juliana Damasio, Vitor Peres, Giovani P Farias, Débora Cristina Engelmann, Jomi Fred Hübner, and Rafael H Bordini. Smart–jacamo: An organisation-based team for the multi-agent programming contest. In *Multi-Agent Progamming Contest*, pages 72–100. Springer, 2018.

[KÕ15] Mare Koit and Haldur Õim. A computational model of argumentation in agreement negotiation processes. *Argument & Computation*, 6(2):101–129, 2015.

[KV11] Diana Kalibatiene and Olegas Vasilecas. Survey on ontology languages. In *International Conference on Business Informatics Research*, pages 124–141. Springer, 2011.

[Lov78] Donald W. Loveland. *Automated theorem proving: a logical basis*, volume 6 of *Fundamental studies in computer science*. North-Holland, 1978.

[LS09] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[MAB+14] Viviana Mascardi, Davide Ancona, Matteo Barbieri, Rafael H Bordini, and Alessandro Ricci. Cool-agentspeak: Endowing agentspeak-dl agents with plan exchange and ontology services. *Web Intelligence and Agent Systems: An International Journal*, 12(1):83–107, 2014.

[MABR11] Viviana Mascardi, Davide Ancona, Rafael H Bordini, and Alessandro Ricci. 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*, volume 2, pages 109–116. IEEE, 2011.

[MB08] Andrés Muñoz and Juan A. Botía. ASBO: argumentation system based on ontologies. In *Cooperative Information Agents XII, 12th International Workshop, CIA 2008, Prague, Czech Republic, September 10-12, 2008. Proceedings*, pages 191–205, 2008.

[Mil19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[MJMW13] Kevin Mackway-Jones, Janet Marsden, and Jill Windle. *Emergency triage: Manchester triage group*. John Wiley & Sons, 2013.

[MP01a] Peter McBurney and Simon Parsons. Chance discovery using dialectical argumentation. In *Annual Conference of the Japanese Society for Artificial Intelligence*, pages 414–424. Springer, 2001.

[MP01b] Peter McBurney and Simon Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11:2002, 2001.

[MP02] Peter McBurney and Simon Parsons. Dialogue games in multi-agent systems. *Informal Logic*, 22:2002, 2002.

[MP05] Peter McBurney and Simon Parsons. Locutions for argumentation in agent interaction protocols. In *Agent Communication*, pages 209–225. Springer, 2005.

[MPR06] Nicolas Maudet, Simon Parsons, and Iyad Rahwan. Argumentation in multi-agent systems: Context and recent developments. In Nicolas Maudet, Simon Parsons, and Iyad Rahwan, editors, *ArgMAS*, volume 4766 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2006.

[MR11] Joaquim Matos and Pedro Pereira Rodrigues. Modeling decisions for hospital bed management - a review. In *4th International Conference on Health Informatics*, pages 504–507, 2011.

[MVBH05a] Alvaro F Moreira, Renata Vieira, Rafael H Bordini, and Jomi F Hübner. Agent-oriented programming with underlying ontological reasoning. In *International Workshop on Declarative Agent Languages and Technologies*, pages 155–170. Springer, 2005.

[MVBH05b] Álvaro F. Moreira, Renata Vieira, Rafael H. Bordini, and Jomi Fred Hübner. Agent-oriented programming with underlying ontological reasoning. In *Declarative Agent Languages and Technologies III, Third International Workshop, DALT 2005, Utrecht, The Netherlands, July 25, 2005, Selected and Revised Papers*, pages 155–170, 2005.

[MZG22] Xin Ma, Xue Zhao, and Pengfei Guo. Cope with the covid-19 pandemic: Dynamic bed allocation and patient subsidization in a public healthcare system. *International Journal of Production Economics*, 243:108320, 2022.

[NM17] Florian Nielsen and Wolfgang Minker. *Assistive and Adaptive Dialog Management*, chapter 9, pages 167–186. Springer International Publishing, Cham, 2017.

[Nut93] Donald Nute. *Defeasible prolog*. Artificial Intelligence Programs, University of Georgia, 1993.

[OEK+22] Juliana Damasio Oliveira, Debora C. Engelmann, Davi Kniest, Renata Vieira, and Rafael H. Bordini. Multi-agent interaction to assist visually-impaired and elderly people. *International Journal of Environmental Research and Public Health*, 19(15), 2022.

[OMD09] Martin O'Connor, Mark Musen, and Amar Das. Using the semantic web rule language in the development of ontology-driven applications. In *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, pages 525–539. IGI Global, 2009.

[ON09] Nir Oren and Timothy J Norman. Arguing using opponent models. In *International Workshop on Argumentation in Multi-Agent Systems*, pages 160–174. Springer, 2009.

[P+20] H Prakken et al. A persuasive chatbot using a crowd-sourced argument graph and concerns. *Computational Models of Argument: Proceedings of COMMA 2020*, 326:9, 2020.

[Pan19] Alison Roberto Panisson. *A Framework for Reasoning and Dialogue in Multi-Agent Systems Using Argumentation Schemes*. Phd thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2019.

[PB16] Alison R Panisson and Rafael H Bordini. Knowledge representation for argumentation in agent-oriented programming languages. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 13–18. IEEE, 2016.

[PB17a] Alison R. Panisson and Rafael H. Bordini. Argumentation schemes in multi-agent systems: A social perspective. In Fallah-Seghrouchni et al. [FRS18], pages 92–108.

[PB17b] Alison R Panisson and Rafael H Bordini. Argumentation schemes in multi-agent systems: A social perspective. In *International Workshop on Engineering Multi-Agent Systems*, pages 92–108. Springer, 2017.

[PB17c] Alison R Panisson and Rafael Heitor Bordini. Uttering only what is needed: Enthymemes in multi-agent systems. In *Proceedings of the 16th International Conference on Autonomous Agents & Multiagent Systems (AAMAS-2017), 2017, Brasil.*, 2017.

[PB20] Alison R. Panisson and Rafael H. Bordini. Towards a computational model of argumentation schemes in agent-oriented programming languages. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2020.

[PBPP14] Alun Preece, Dave Braines, Diego Pizzocaro, and Christos Parizas. Human-machine conversations to support multi-agency missions. *ACM SIGMOBILE Mobile Computing and Communications Review*, 18(1):75–84, 2014.

[PEB21] Alison R Panisson, Débora C Engelmann, and Rafael H Bordini. Engineering explainable agents: an argumentation-based approach. In *International Workshop on Engineering Multi-Agent Systems*, pages 273–291. Springer, 2021.

[PFF+14] Alison R Panisson, Giovani Farias, Artur Freitas, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Planning interactions for agents in argumentation-based negotiation. In *Proc. of 11th Int. Workshop on Argumentation in Multi-Agent Systems*, pages 1–15, 2014.

[PGB03] N. C. Proudlove, K. Gordon, and R. Boaden. Can good bed management solve the overcrowding in accident and emergency departments? *Emergency Medicine Journal*, 20(2):149–155, 2003.

[PM03] Simon Parsons and Peter McBurney. Argumentation-based dialogues for agent co-ordination. *Group Decision and Negotiation*, 12(5):415–439, 2003.

[PMB21] Alison R Panisson, Peter McBurney, and Rafael H Bordini. A computational model of argumentation schemes for multi-agent systems. *Argument & Computation*, (Preprint):1–39, 2021.

[PMF+14] Alison R. Panisson, Felipe Meneguzzi, Moser Silva Fagundes, Renata Vieira, and Rafael H. Bordini. Formal semantics of speech acts for argumentative dialogues. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1437–1438, 2014.

[PMVB14] Alison R Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. An approach for argumentation-based reasoning using defeasible logic in multi-agent programming languages. In *11th International Workshop on Argumentation in Multiagent Systems*, pages 1–15, 2014.

[PMVB15a] Alison R Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Towards practical argumentation-based dialogues in multi-agent systems. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 151–158. IEEE, 2015.

[PMVB15b] Alison R. Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H. Bordini. Towards practical argumentation in multi-agent systems. In *2015 Brazilian Conference on Intelligent Systems, BRACIS 2015*, pages 98–103, 2015.

[Pra00] Henry Prakken. On dialogue systems with speech acts, arguments, and counterarguments. In *European Workshop on Logics in Artificial Intelligence*, pages 224–238. Springer, 2000.

[Pra11] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2011.

[Pru11] Jaume Jordàn Prunera. Case-based argumentation infrastructure for agent societies. Master's thesis, Universitat Politècnica de València, Espanha, Sep 2011.

[PSM+18a] Alison R. Panisson, Stefan Sarkadi, Peter McBurney, Simon Parsons, and Rafael H. Bordini. On the formal semantics of theory of mind in agent communication. In *Agreement Technologies - 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers*, pages 18–32, 2018.

[PSM+18b] Alison R Panisson, Stefan Sarkadi, Peter McBurney, Simon Parsons, Rafael H Bordini, et al. Lies, bullshit, and deception in agent-oriented programming languages. In *CEUR Workshop Proceedings*, volume 2154, pages 50–61, 2018.

[PWA02] Simon Parsons, Michael Wooldridge, and Leila Amgoud. An analysis of formal inter-agent dialogues. In *In 1st International Conference on Autonomous Agents and Multi-Agent Systems*, pages 394–401. ACM Press, 2002.

[RA06] Iyad Rahwan and Leila Amgoud. An argumentation-based approach for practical reasoning. In Nicolas Maudet, Simon Parsons, and Iyad Rahwan, editors, *ArgMAS*, volume 4766 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 2006.

[RAMI17] AM Rahman, Abdullah Al Mamun, and Alma Islam. Programming challenges of chatbot: Current and future prospective. In *Humanitarian Technology Conference (R10-HTC), 2017 IEEE Region 10*, pages 75–78. IEEE, 2017.

[RBR+19] Geetanjali Rakshit, Kevin K Bowden, Lena Reed, Amita Misra, and Marilyn Walker. Debbie, the debate bot of the future. In *Advanced Social Interaction with Agents*, pages 45–52. Springer, 2019.

[RG+95] Anand S Rao, Michael P Georgeff, et al. Bdi agents: from theory to practice. In *Icmas*, volume 95, pages 312–319, 1995.

[RPVO09] Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini. Environment programming in cartago. In *Multi-agent programming*, pages 259–288. Springer, 2009.

[RR19] Avi Rosenfeld and Ariella Richardson. Explainability in human–agent systems. *Autonomous Agents and Multi-Agent Systems*, 33(6):673–705, 2019.

[RTO13] Tjitze Rienstra, Matthias Thimm, and Nir Oren. Opponent models with uncertainty for strategic argumentation. In *IJCAI 2013, Proceedings of the 23rd*

International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, pages 332–338, 2013.

[RWP+18] Niklas Rach, Klaus Weber, Louisa Pragst, Elisabeth André, Wolfgang Minker, and Stefan Ultes. Eva: a multimodal argumentative dialogue system. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, pages 551–552, 2018.

[SA07] Bayan Abu Shawar and Eric Atwell. Chatbots: are they really useful? In *LDV Forum*, volume 22, pages 29–49, 2007.

[SBMV15] Daniela Schmidt, Rafael H Bordini, Felipe Meneguzzi, and Renata Vieira. An ontology for collaborative tasks in multi-agent systems. In *Ontobras*, volume 1442, 09 2015.

[SIH+18] Kazuki Sakai, Akari Inago, Ryuichiro Higashinaka, Yuichiro Yoshikawa, Hiroshi Ishiguro, and Junji Tomita. Creating large-scale argumentation structures for dialogue systems. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[SPB+18] Ştefan Sarkadi, Alison R Panisson, Rafael H Bordini, Peter McBurney, and Simon Parsons. Towards an approach for modelling uncertain theory of mind in multi-agent systems. In *International Conference on Agreement Technologies*, pages 3–17. Springer, 2018.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51 – 53, 2007. Software Engineering and the Semantic Web.

[STT01] Fariba Sadri, Francesca Toni, and Paolo Torroni. Logic agents, dialogues and negotiation: An abductive approach. In *Proceedings of the AISB 2001 Symposium on Information Agents for E-Commerce*, pages 1–8. AISB, 2001.

[TEDF+12] K. L. Teow, E. El-Darzi, C. Foo, X. Jin, and J. Sim. Intelligent analysis of acute bed overflow in a tertiary hospital in singapore. *Journal of Medical Systems*, 36:1873–1882, June 2012.

[TJ20] Sandeep A Thorat and Vishakha Jadhav. A review on implementation issues of rule-based chatbot systems. *Available at SSRN 3567047*, 2020.

[TL08] Quynh-Nhu Numi Tran and Graham Low. Mobmas: A methodology for ontology-based multi-agent systems development. *Information and Software Technology*, 50(7):697 – 722, 2008.

[TM11] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer, 2011.

[Ton14] Francesca Toni. A tutorial on assumption-based argumentation. *Argument and Computation*, 5-1:89–117, 2014.

[Tou58] St Toulmin. The uses of argument cambridge university press. *Cambridge, UK*, 34, 1958.

[TPW+16] Alice Toniolo, Alun D Preece, Will Webberley, Timothy J Norman, Paul Sullivan, and Timothy Dropps. Conversational intelligence analysis. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, pages 1–6, 2016.

[TYZ12] Xuehong Tao, Nicola Yelland, and Yanchun Zhang. Fuzzy cognitive modeling for argumentative agent. In *2012 IEEE International Conference on Fuzzy Systems*, pages 1–8. IEEE, 2012.

[VASS05] R. Vieira, D. S. Abdalla, D. M. Silva, and M. R. Santana. *Web Semântica: Ontologias, Lógica de Descrição e Inferência*, volume 1, pages 127–167. SBC, 2005.

[VGBM17] Mitchell Vibbert, Jacques-Olivier Goussard, Richard J Beaufort, and Benjamin P Monnahan. Dialog flow management in hierarchical task dialogs, September 19 2017. US Patent 9,767,794.

[VMWB07] Renata Vieira, Álvaro F. Moreira, Michael J. Wooldridge, and Rafael H. Bordini. On the formal semantics of speech-act based communication in an agent-oriented programming language. *J. Artif. Intell. Res.*, 29:221–267, 2007.

[Wal10] Douglas Walton. Types of dialogue and burdens of proof. volume 216, pages 13–24, 01 2010.

[Wal19] Douglas Walton. Using argumentation schemes to find motives and intentions of a rational agent. *Argument & Computation*, (Preprint):1–43, 2019.

[WJR+20] Klaus Weber, Kathrin Janowski, Niklas Rach, Katharina Weitz, Wolfgang Minker, Stefan Ultes, and Elisabeth André. Predicting persuasive effectiveness for multimodal behavior adaptation using bipolar weighted argument graphs. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS '19, Auckland, New Zealand, May, 2020*, page 1476–1484, 2020.

[WK95] Douglas Walton and Erik CW Krabbe. *Commitment in dialogue: Basic concepts of interpersonal reasoning*. SUNY press, 1995.

[Woo00] Michael J. Wooldridge. Computationally grounded theories of agency. In *4th International Conference on Multi-Agent Systems, ICMAS 2000, Boston, MA, USA, July 10-12, 2000*, pages 13–22. IEEE Computer Society, 2000.

[Woo09] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[WRM08] Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation schemes*. Cambridge University Press, 2008.

[WY16] Xiaojie Wang and Caixia Yuan. Recent advances on human-computer dialogue. *CAAI Transactions on Intelligence Technology*, 1(4):303 – 312, 2016.

[WYAL19] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. Designing theory-driven user-centric explainable ai. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15, 2019.

[YJ95] L Richard Ye and Paul E Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *Mis Quarterly*, pages 157–172, 1995.

[ZEJ+22] Chi Zhang, Torsten Eken, Silje Bakken Jørgensen, Magne Thoresen, and Signe Søvik. Effects of patient-level risk factors, departmental allocation and seasonality on intrahospital patient transfer patterns: network analysis applied on a norwegian single-centre data set. *BMJ open*, 12(3):e054545, 2022.