ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ALISON ROBERTO PANISSON

A FRAMEWORK FOR REASONING AND DIALOGUE IN MULTI-AGENT SYSTEMS USING ARGUMENTATION SCHEMES

Porto Alegre
2019

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

A FRAMEWORK FOR REASONING AND DIALOGUE IN MULTI-AGENT SYSTEMS USING ARGUMENTATION SCHEMES

ALISON ROBERTO PANISSON

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Dr. Rafael Heitor Bordini

Porto Alegre
2019

Ficha Catalográfica

P192f Panisson, Alison Roberto

A framework for reasoning and dialogue in multi-agent systems using argumentation schemes / Alison Roberto Panisson . – 2019.
149.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Heitor Bordini.

1. Multi-Agent Systems. 2. Argumentation. 3. Argumentation Schemes. 4. Enthymemes. I. Bordini, Rafael Heitor. II. Título.

Alison Roberto Panisson

A Framework for Reasoning and Dialogue in Multi-Agent Systems using Argumentation Schemes

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 13, 2019.

COMMITTEE MEMBERS:

Prof. Dr. Avelino Francisco Zorzo (PPGCC/PUCRS)

Profa. Dr. Cesar Augusto Tacla (UTFPR)

Prof. Dr. Luís Alvaro de Lima Silva (UFSM)

Prof. Dr. Rafael Heitor Bordini (PPGCC/PUCRS - Advisor)

I dedicate my work to my family and friends.

“Argumentation constitutes a major component of human intelligence”
(Plan Minh Dung [45])

ACKNOWLEDGMENTS

I would like to thank my family and friends, in special my wife Duda, for the support and help received during this journey. Family and friends are the most important things in my life, surely they are the ones who give us the strength to pursue our dreams and goals. I would like to cite my friend's name Marciel Parizzoto, who passed away during the time I was doing my Ph.D in the UK and I could not say goodbye to him. It remembers me that we can follow only one path guided by our choices and, consequently, many of those choices will make us miss many moments with those people we love, that is why we should give the best of us, making we worth the time we have.

I would like to thank the group titled "Semantic, Multi-Agent, and Robotic Technologies" and all its members for the help and encouragement to my research, in particular, I would like to express my sincere gratitude to my advisor Prof. Dr Rafael H. Bordini for the continuous support of my Ph.D study and research, for his encouragement and trust on my work, for his patience, motivation, enthusiasm, and immense knowledge. I could not have imagined having a better advisor and mentor for my Ph.D study.

I would like to thank Prof. Dr Peter McBurney for his supervision during the time I expended at the King's College London, from September 2017 to March 2018. Prof. McBurney is an excellent advisor and mentor, he always had an excellent reference to support my ideas. Also, I would like to thank Prof. Dr Simon Parsons, Dr Asad Ali, and other members of the King's College London for the meetings and help with my research.

Finally, I would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brasil (CAPES) for funding my research. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brasil (CAPES) – Finance Code 001

UM FRAMEWORK PARA RACIOCÍNIO E DIÁLOGO EM SISTEMAS MULTIAGENTES USANDO ESQUEMAS DE ARGUMENTAÇÃO

RESUMO

Um dos aspectos mais importantes em sistemas multiagentes é a comunicação. Entre as técnicas de comunicação em sistemas multiagentes, abordagens baseadas em argumentação tem recebido um interesse especial por parte da comunidade dado que essas abordagens fornecem uma rica forma de comunicação permitindo agentes comunicarem argumentos. Além de uma rica forma de comunicação, argumentação também fornece um forma elegante de raciocínio não monotonico. Usando argumentação, agentes são capazes de alcançar conclusões bem suportadas, construindo argumentos a favor e contra suas atitudes mentais.

Para se comunicar e raciocinar usando argumentação, agentes precisam instanciar argumentos dos padrões de raciocínio (esquemas de argumentação) disponíveis para eles. No entanto, não há uma abordagem geral para representar esquemas de argumentação em sistemas multiagentes. Assim, nesta tese, investigamos como esquemas de argumentação podem ser representados em sistemas multiagentes e como agentes podem instanciar esses padrões de raciocínio para raciocinar e se comunicar. Baseado nessa investigação, propomos um *framework* para esquemas de argumentação em sistemas multiagentes.

Além disso, as informações adicionais trocadas por agentes durante comunicações baseadas em argumentação podem sobrecarregar a infraestrutura de comunicação, restringindo a aplicação de técnicas de argumentação na prática. Assim, baseados na estrutura para esquemas de argumentação em sistemas multiagentes proposta, também propomos uma abordagem pela qual os agentes podem trocar mensagens mais curtas durante diálogos, omitindo informações que são de conhecimento comum entre os agentes (por exemplo, informações sobre a organização multiagente compartilhada). Em particular, nossa abordagem usa a ideia de *enthymemes*, bem como referências para esquemas de argumentação compartilhados e conhecimento organizacional compartilhado para guiar a reconstrução de argumentos.

Palavras-Chave: Sistemas Multiagentes, Argumentação, Esquemas de Argumentação, *Enthymemes*.

A FRAMEWORK FOR REASONING AND DIALOGUE IN MULTI-AGENT SYSTEMS USING ARGUMENTATION SCHEMES

ABSTRACT

One of the most important aspects of multi-agent systems is communication. Among the communication techniques in multi-agent systems, argumentation-based approaches have received special interest from the research community because those approaches provide a rich form of communication by means of agents exchanging arguments. Besides a rich form of communication, argumentation also provides an elegant form of nonmonotonic reasoning. Using argumentation, agents are able to reach well-supported conclusions, constructing arguments for and against their mental attitudes.

In order to communicate and reason using argumentation, agents need to instantiate arguments from the reasoning patterns (argumentation schemes) available to them. However, there is no general approach to represent argumentation schemes in multi-agent systems. Thus, in this thesis, we investigate how argumentation schemes can be represented in multi-agent systems, and how agents can instantiate those reasoning patterns in order to reason and communicate. Based on such an investigation, we propose a framework for argumentation schemes in multi-agent systems.

Furthermore, the additional information exchanged by agents during argumentation-based communication can overburden the communication infrastructure, restricting the practical applications of argumentation techniques. Thus, based on our framework for argumentation schemes in multi-agent systems, we also propose an approach whereby agents are able to exchange shorter messages when engaging in dialogues by omitting information that is common knowledge (e.g., information about a shared multi-agent organisation). In particular, our approach uses the idea of *enthymemes* as well as referring to shared argumentation schemes and common organisational knowledge to guide argument reconstruction.

Keywords: Multi-Agent Systems, Argumentation, Argumentation Schemes, Enthymemes.

CONTENTS

1. INTRODUCTION

The theory of argumentation is a rich interdisciplinary area of research in philosophy, communication studies, linguistics, computer science, and psychology [156]. In the past few years, formal models of argumentation started to stand out and gained importance in artificial intelligence, given they have found a wide range of application in specifying semantics for logic programmings, generating natural language text, supporting legal reasoning, and facilitating multi-agent dialogues and negotiation on the Internet [156].

In particular, argumentation has received significant interest in the Multi-Agent System community in recent years [17, ?, 68, 72, 78, 89, 114, 131, 142, 145, 150]. The two main lines of research in the multi-agent community regarding argumentation are argumentation-based reasoning and argumentation-based communication [78]. Argumentation-based reasoning has a focus on agents reasoning over incomplete, conflicting, or uncertain information (nonmonotonic reasoning), where they are able to construct arguments for and against certain conclusions (beliefs, goals, etc.) of their own. Comparing such arguments for and against their conclusions, agents are able to reach acceptable conclusions, given the knowledge available to them. On the other hand, argumentation-based communication has a focus on communication between agents, allowing the exchange of arguments to justify a stance and to provide reasons that defend claims made by individual participants. The main claim appearing in the literature about the benefits of an argumentation-based approach to communication is the fact that this type of communication allows agents to reach agreements in situations where other approaches would not (for example, in negotiation, where argumentation is compared to game-theoretic and heuristic-based approaches [126]). Such benefits are the consequence of the exchange of additional information (arguments) which allow agents to make more informed decisions and comparisons.

Although argumentation could have an important role in multi-agent systems, most of the work in the field of argumentation is concentrated in properties of theoretic frameworks describing the interaction of arguments, most of them based on abstract argumentation [45]. Closer to the needs of multi-agent systems, the studies in structured argumentation aim to give structure to arguments, allowing arguments to be interpreted according to the meaning of their contents. Such approaches to structured arguments allow us to use different logical languages, normally based on different kinds of inference rules. Even within the studies in structured argumentation there is little work that investigates argumentation in the context of agent-oriented programming languages, or even in multi-agent platforms/frameworks, which could enable extensive use of argumentation techniques in practical multi-agent systems development. Such an investigation would include, for example, a particular language to represent knowledge/arguments, which could be naturally interpreted by agents, a communication language defining precisely the exchange of arguments among agents in such systems, and it could include the consideration of particular concepts present in multi-agent systems, such as organisational structures. This thesis aims to do exactly that.

Further, although approaches for structured argumentation provide a manner to generate and evaluate the acceptability of arguments from a logical representation, which arguments agents will be able to construct depends on the knowledge available to them and on inference patterns (or reasoning patterns) available for that particular domain/application. Towards defining reasoning patterns in multi-agent systems, argumentation schemes [156] have been discussed in this context [145, 113, 142]. Argumentation schemes are reasoning patterns widely known in argumentation studies, describing forms of arguments (structure of inferences) that represent structures of reasoning used in everyday discourse (i.e., common types of arguments used in everyday discourse), as well as specific contexts such as legal argumentation and scientific argumentation [156].

Towards exploring the potential of argumentation techniques in multi-agent systems, in this document we present our research that focused on bringing to multi-agent systems much of the studies done in argumentation. In particular, we are interested in combining studies done in argumentation with multi-agent oriented programming paradigm (e.g., the JaCaMo Framework [31]).

In order to define reasoning patterns in multi-agent systems, we investigated different approaches for argumentation schemes, identifying a common structure for argumentation schemes. Based on this common structure, we were able to propose a formal specification for argumentation schemes in multi-agent system [103, 102, 99]. This was an important step in our research, given that, although argumentation schemes are widely studied reasoning patterns used by humans in everyday discourse, there are few studies identifying and using argumentation schemes in the context of multi-agent systems. Also, the role of argumentation schemes in the multi-agent oriented programming paradigm is unclear. Further, considering argumentation schemes in multi-agent systems from a practical point of view seems as yet an underdeveloped research topic. Thus, based on the formal specification for argumentation schemes, we suggest an infrastructure in which agents share argumentation schemes, they are able to instantiate arguments from theses reasoning patterns, and they are able to use the multiples instances of arguments during reasoning and communication. Our approach is built based on the multi-agent oriented programming paradigm, resulting in a complete synergy between the formal specification of our framework and its implementation.

In chapter 3, we present our research regarding the structure and representation of argumentation schemes in multi-agent systems. We adopt a view based on the idea that domain specific knowledge is shared by agents in multi-agent systems [49]. Therefore, we propose that argumentation schemes could be shared in multi-agent systems as well, considering that they describe common reasoning patterns which will be available for agents to instantiate arguments in that particular system/society. In order to represent and share argumentation schemes in multi-agent systems, and particularly in the multi-agent oriented programming paradigm, we proposed two alternatives ways: (i) in Section 3.1, we propose a model for argumentation schemes extending a multi-agent organisation with a new dimension for argumentation schemes [102]; and (ii) in Section 3.2, we propose an infrastructure for sharing argumentation schemes using semantics data bases (ontologies) [49, 50]. In both approaches, agents are directly able to instantiate such argumentation schemes, similarly to the way they are aware of the organisation structure.

Agents are able to instantiate arguments from the argumentation schemes and use those instantiated arguments during reasoning and communication. In Chapter 4, we present our research regarding argumentation-based reasoning. We have developed an argumentation-based reasoning mechanism to enable agents to reason, generate, and evaluate acceptable arguments in their reasoning processes [101, 109]. Such an argumentation-based reasoning mechanism was the first step towards the goal of this research, considering that agents need to be able to interpret and reason about arguments they receive during communication, as well as they need to be able to generate and evaluate arguments for and against their mental attitudes (beliefs, intentions, goals) in order to reach acceptable conclusions and to engage in meaningful communication. Similar to most of the structured approaches to argumentation, the reasoning mechanism we developed works by using inference rules representing reasoning patterns (normally specified for particular applications domains), which was extended to "linked predicates" in order to consider a richer structure of argumentation schemes. In Chapter 4, we start presenting, in Sections 4.1 and 4.2, the basic argumentation-based reasoning mechanism we have developed [109, 101]. Afterwards, in Section 4.4, we present its extended version considering the richer structure of argumentation schemes. Finally, in Section 4.5, we evaluate our approach, showing its generality and performance in argumentation-based reasoning.

In Chapter 5, we present our research regarding argumentation-based dialogues. In Section 5.1, we start presenting the operational semantics for a set of speech acts for argumentation in agent-oriented programming languages based on the BDI (Beliefs-Desires-Intentions) architecture we have proposed in [111, 108]. In Section 5.2, we present the modules and artifacts we have implemented in order to enable argumentation-based dialogues in multi-agent systems. In Section 5.4, we discuss the role of argumentation schemes in argumentation-based dialogues. In Section 5.5, we evaluate our framework for argumentation-based dialogues considering argumentation schemes. In order to evaluate our framework, we define two different protocols for argumentation-based dialogues: (i) a protocol for persuasion, based on our previous work [110, 106, 136]; and (ii) a protocol for data access control in multi-agent systems [100]. After we implemented those protocols, we ran some simulation. Again, we show the generality of our framework, as it supports the implementation of different protocols. Finally, in Section 5.6, we present an approach to enthymemes in multi-agent systems [103]. As it will be easily noted, argumentation-based communication makes agent communication richer. However, it leads to overheads in agent communication, given the extra content used in each interaction. Considering that some multi-agent applications could have restricted network availability, e.g., mobile applications, such computational costs should be avoided as much as possible. Enthymemes are arguments that can have some of its parts omitted because of the knowledge shared by the proponent and recipient of such an argument. Therefore, that allows agents to exchange only the relevant information to understand each other. Thus, in our approach for enthymemes in multi-agent systems, we propose agents to exchange the minimum possible content needed for a mutual understanding of the communication. The infrastructure we have proposed for sharing argumentation schemes in multi-agent systems seems to encourage the use of enthymemes in multi-agent systems, guaranteeing the efficiency required in agent communication.

2. BACKGROUND

2.1 Agent-Oriented Programming Languages

In the agent-oriented programming paradigm, the agents are computational entities with autonomous behaviour (*i.e.*, able to make decisions and act without direct human intervention on unexpected circumstances). These computational entities are situated in an environment that they are able to sense (through sensors), act upon it (through effectors), and communicate through message passing [164].

One of the most studied architectures for cognitive agents is the BDI (*Beliefs-Desires-Intentions*) architecture which provides a particular structure for agent internal states based on "mental attitudes". The internal state of a BDI agent is formed by: (i) *Beliefs* that represent the information about the world (including itself and other agents) available to that agent; (ii) *Desires* representing the motivations of the agent, *i.e.*, the states of the environment that the agent would like to reach; and (iii) *Intentions* which are desires that the agent is committed to achieve by following particular *plans* of action.

There exist many agent-oriented programming languages and platforms, such as Jason, Jadex, Jack, AgentFactory, 2APL, GOAL, Golog, and MetateM, as pointed out in [32]. Those languages differ in the agent architecture used, in the form of communication/interaction between them, and also on the programming paradigms that inspired or underlie each language.

Among the languages mentioned above, AgentSpeak(L), the language on which Jason [33] is based, is one of the best-known languages inspired by the BDI architecture. AgentSpeak(L) is an abstract logic-based agent-oriented programming language introduced by Rao [128], and subsequently extended and formalised in a series of papers by Bordini, Hübner, and various colleagues. Also, Jason is part of the complete framework to develop multi-agent systems applications considering all the dimensions we need, named JaCaMo [31] (which we describe in Section 2.4).

AgentSpeak(L) is based on the Procedural Reasoning System (PRS) (Figure 2.1) where the agents are equipped with a library of pre-compiled plans. Plans in PRS have the following components: (i) a *goal* – the post-condition of the plan (the things that it achieves); (ii) a *context* – the pre-condition for the plan, defining what must be true of the environment in order for the plan to be successful; and (iii) a *body* – the 'recipe' part of the plan – it may contain a list of actions and sub-goals in order to achieve the main goal.

Plans in AgentSpeak(L) have the following format:

```
triggering_event : context <- body.
```

where the `triggering_event` represents a new agent goal (or belief), which is to be pursued and has the format !`goal(Parameter)`, the `context` has preconditions needed to perform that plan to achieve that goal, and the `body` is a sequence of actions and sub-goals (which trigger others events and the use of other plans) to achieve the goal.

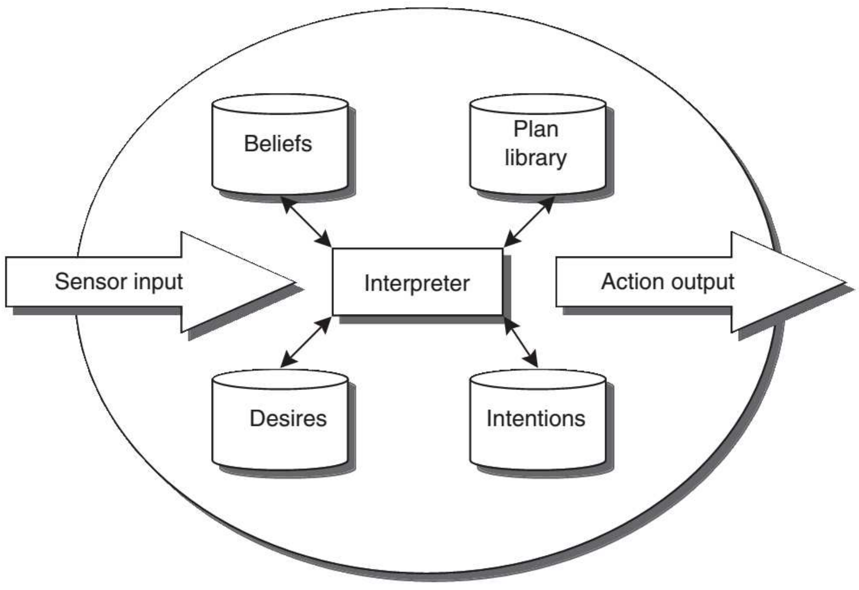


Figure 2.1 – The Procedural Reasoning System (PRS) ([33]).

In particular, the main AgentSpeak(L) extensions available in Jason (a Java-based platform for the development of multi-agent systems), according to [33] are:

- **Strong negation**: Strong negation helps the modeling of systems where uncertainty cannot be avoided, allowing the representation of things that the agent believes to be true, believes to be false, and things that the agent is ignorant about;

- **Handling of plan failures**: Jason has a particular form of plan failure handling mechanism consisting of plans that are triggered by failure, giving the programmer the chance to act so as to undo the effects of any action already done before the plan failed, if necessary, and then adopting the goal (that was not achieved) again, if the conditions are appropriate.

- **Belief annotations**: One interesting characteristic present in Jason is that it automatically generates annotations for all beliefs in the belief base about the source from where the belief was obtained (sensing the environment, communication with other agents, or a mental note created by the agent itself). The annotation has the following format: `likes(john, music)[source(john)]`, stating that the source of the belief that john likes music is agent john itself.

- **Speech-act based communication**: Jason uses performatives based on speech acts in its communication language, which goes well with the availability of formal semantics of mental attitudes for the Jason extension of AgentSpeak;

- **Plan annotations**: programmers can add annotations to plan labels which can be used by elaborate selection functions, for example for defining preferences in case various different plans are applicable.

The Jason platform, more generally, has the following features:

- **Distribution**: The platform permits easy distribution on a computer network; further details can be found in [33];
- **Environment**: In many cases, during development, a simulation of the target application environment will be needed, so Jason provides support for developing environments, which are programmed in Java;
- **Customisation**: Programmers can customise important parts of the agent platform by providing application-specific Java methods for certain aspects of an agent and the agent architecture. These parts can be customised through methods of the agent class and agent architecture class (more details in [33]);
- **Language extensibility and legacy code**: The extension of the AgentSpeak language in Jason can be done through the so called "internal actions", which are implemented in Java (or any other language using Java Native Interface);
- **Integrated Development Environment**: Jason is distributed as a plug-in for the jEdit and Eclipse IDEs (Integrated Development Environment). This facilitates the development of applications.

As mentioned above, the communication between agents is via message passing, and Jason's communication is based on speech acts and is performed by the pre-defined internal action '`.send`', of the following format:

```
.send(receiver, illocutionary_force, propositional_content)
```

where `receiver` is the name of an agent (each agent has a unique individual name in the multi-agent system) or a list of agent names, for whom the message is being sent. The `propositional_content` is a term in AgentSpeak (a literal, triggering event, plan, or a list of literals or plans). The `illocutionary_force` denotes the intention of the sender (often called *performative*), as in the speech-act theory. The formal semantics of receiving such messages is given in [151], and a complete list of all the illocutionary forces available can be found in [33].

New illocutionary forces can be added, as well as the effects that each will have on the agent's mental state. This is an important feature for our work, because argumentation-based dialogues need illocutionary forces that represent specific intentions of message senders, as well as the required effect in the mental state of the receiver of the message. In Jason, agent plans can be written in AgentSpeak to give such semantics to new performatives, hence providing an elegant and transparent way for programming agents that are capable of such argumentations.

2.2 Organisation in Multi-Agent Systems

"Multi-Agent Systems have evolved towards the specification of global constraints that heterogeneous and autonomous agents are supposed to follow when concerning open systems. A subset of these constraints is known as the multi-agent systems organisation" [66].

In open multi-agent systems, the autonomy of the agents can lead the overall system to undesired behaviour, since each agent does what it wants. This problem may be solved by assigning an organisation to the system [66]. An organisation can be considered as a set of behavioural constraints which the agent is subjected when entering into the system, assuming a role in its organisation [66].

We can have an explicit representation of the organisation available to the agents at runtime and it is desirable that the agents been able to read, represent, and reason about the organisation. In these kinds of systems the agents can exert organisational autonomy, since they may decide whether to follow the organisation or not [66]. To develop multi-agent systems with this characteristics we need of a framework to ensure organisational constraints and an agent-oriented programming language that supports the agent decision making about its organisation, for example, Jason Platform [33] and Moise[1] [65].

The Moise model has three dimensions [66]:

- The *functional dimension*: which refers to the specification of global plans and goals. The functional dimension is composed by a set of schemes which represent how multi-agent systems usually achieve its *global* (organisational) goals stating how these goals are decomposed (by plans) and distributed to the agents (by missions). Schemes can be seen as a goal decomposition tree where the root is the global goal, and the leafs are goals that can be achieved by the agents.

- The *structural dimension*: which refers to the roles, the relation between the roles and the group of roles. That is, the structural dimension is built in three levels:

  - *individual level*: which defines the behaviours that an agent is responsible for when it adopts a role;
  - *social level*: which defines the acquaintance, communication, and authority links between roles;
  - *collective level*: which defines the aggregation of roles in groups;

- The *normative dimension*: which refer to the norms that specify the obligation and permissions linked to each role into the organisation. Roles are indirectly linked to missions by means of permissions and obligations. Roles, also, add some independence between the functional and the structural specifications.

[1]Moise is also part of the JaCaMo Framework that we describe later.

The MOISE model [66] allows that the *normative* and *structural* dimensions can be changed without changing the *functional* dimension, so the agents can change its roles and the norms and achieve the global goals, likewise.

Therefore, the MOISE model [66] enables the declaration of multi-agent systems organisational structures (role, groups, links), functioning (global goals, global plans, missions), obligations, and permissions. Also, the approach is suitable for reorganisation where the specification of the organisation can dynamically change. Such characteristics play an important role in our research, as we will describe in Section 3. The Moise model provides organisational information to agents, as well as allowing us to have more control regarding communication in multi-agent systems. While to have more control regarding communication in multi-agent systems is an interesting topic of research, in this work we are going to focus on the how agents use the organisational information provided by the moise model.

For example, recently, in [63, 30], it was put forward that communication in multi-agent systems could be defined in the organisation of such systems. The first work, [63], proposed a new dimension called *dialogic* for the Moise model; such a dimension focuses on the communication between roles. The authors argue that communication is one of the main tools that agents have to coordinate their actions at the social level, the specification of the exchanges/communications between roles could be a tool for the regulation of the exchanges between agents. Therefore, they suggest the specification of communication between roles (which agents adopt in the organisation) at the organisational level. The *dialogic* dimension, extending the Moise model, is inspired by the theoretical PopOrg model [38, 39]. The second work, [30], introduced the idea that the normative specification should include also the control of communication modes in the organisation. Therefore, the authors proposed to extend the normative part of the organisational model of Moise in order to specify the interaction modes between agents participating within the organisation. That work aims to allow the multi-agent organisation to monitor the interaction between agents, and to make the agents able to reason over the communication modes, similarly to the way they do with norms. We take some inspiration from that work in our research, extending the Moise model with a new dimension for argumentation schemes, and describing the relations among the four dimensions resulting from such extension. That research is presented in Section 3.

2.3 Environment in Multi-agent Systems

Environments for multi-agent systems are abstractions of real or virtual environments, offering mechanisms for agents to manipulate and perceive the environment in which they are situated. Multi-agent applications developed using Jason, normally use the CArtAgO Platform [132] for creating agent environments, given the most recent multi-agent framework called JaCaMo [31]. CArtAgO is a platform based on the *artifact* notion in multi-agent systems. Artifacts are function-oriented computational abstractions which provide services that agents can exploit to support their activities. An artifact makes its functionalities available and exploitable by agents through a set of operations and a set of observable properties [132]. Operations represent computational processes executed inside artifacts, that may be triggered by agents or other artifacts.

Recently, artifacts have been used as control structures for communication, e.g., [94] and our previous work [97]. In our previous work, CArtAgO artifacts are used to store agent commitments regarding what they say in argumentation-based dialogues. More details will be described in Section 5.

2.4 The JaCaMo Framework – All Dimensions in a Single Framework

The JaCaMo Framework [31] brings to multi-agent systems a perspective to the programming level, demonstrating a concrete programming model and platform that allows the integration of the three multi-agent programming dimensions described in the previous sections, i.e., agents, organisation, and environment levels. Also, JaCaMo has been described as the first successful combination of the agent-oriented programming (AOP), organisational-oriented programming (OOP) and environment-oriented programming (EOP) paradigms, proposing a full multi-agent oriented programming (MAOP) paradigm [31]. Despite the proposal for the unified JaCaMo Framework [31] (often also referred to as JaCaMo model) the authors agree that there is much more research in the separated areas, which compose the framework, still to be done; however, the JaCaMo framework is one of the first steps towards the development of complex (social) autonomous systems based on a multi-agent paradigm combining all the clearly necessary dimensions.

As described in the literature, agent programming platforms (Jason, 2APL, GOAL, Jack, JADE, etc.) are able to deal mostly with the agent level, thus some high-level concepts, considered in the conception of multi-agent systems, such as organisational and environment concepts cannot be directly mapped. Commonly, such concepts are mapped as agents as well, in order to provide an alternative solution for such a gap between the conceptual and practical modelling [31].

A JaCaMo multi-agent system (i.e., a software system programmed in JaCaMo) results from an agent organisation programmed in Moise [65], organising autonomous agents programmed in Jason [33], working in a shared distributed artifact-based environments programmed in CArtAgO [132], as it is shown in Figure 2.2.

In Figure 2.3, it is possible to observe the relation/dependencies between components of the different dimensions. The connections terminating with filled squares, in Figure 2.3, represent the synergies and the conceptual mapping between components of the conceptual components in different dimensions. According to the authors of JaCaMo [31], the mapping was produced transparently and without extra programming effort. The connection links between the Agent and Environment dimensions is given by a semantical mapping of agents' external actions into artifact operations and artifact observable properties, and events into agent percepts. This means that agents will be able to execute a particular action in the environment if there is (at least) one artifact providing such corresponding operation for that action [31]. The connections with non-

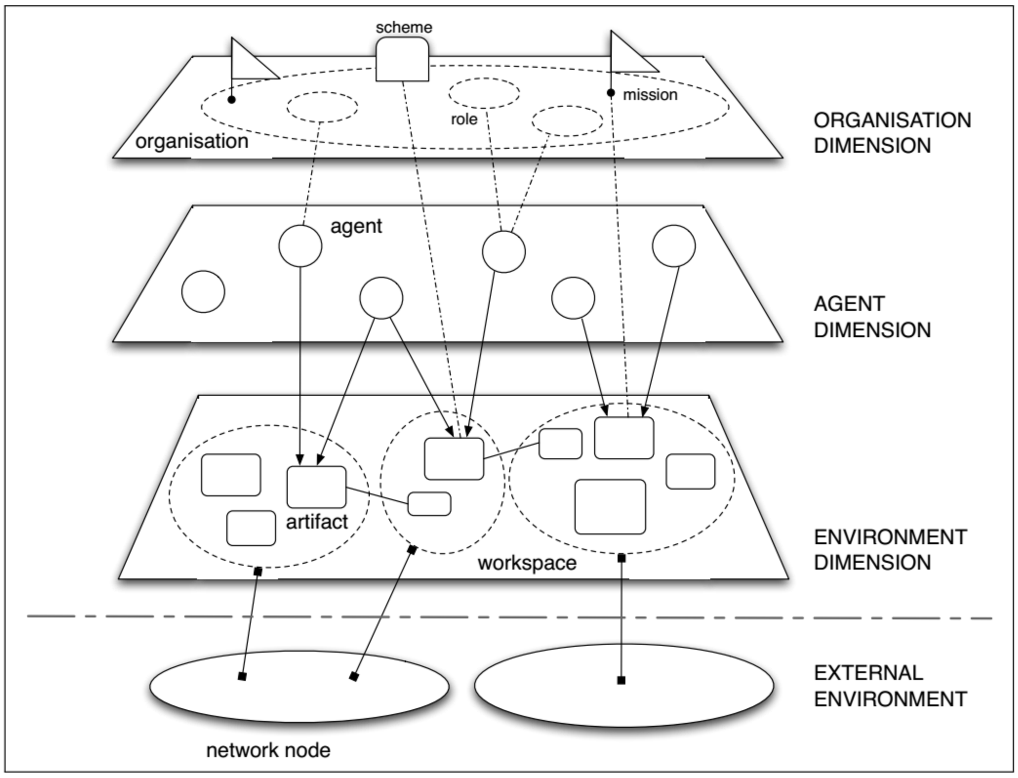


Figure 2.2 – Overview of a JaCaMo multi-agent system [31].

filled squares represent a set of predefined actions that agents can perform, which refer to the basic functionalities provided by the overall infrastructure, including the environment and organisational dimensions [31].

Therefore, JaCaMo [31] provides us with a complete framework for developing multi-agent systems, including the three main dimensions necessary in such a development: agents, organisation, and environment. Although, JaCaMo is one of the most complete frameworks for developing multi-agent systems, and there is a recent investigation regarding interaction in JaCaMo Framework presented in [167], there is no work regarding argumentation in the JaCaMo Framework which explores all those dimensions. Of course, our proposal is not concerned with JaCaMo solely, but as JaCaMo provides us a complete framework for developing multi-agent systems and encompasses much of the research in the multi-agent programming paradigm, it is one of the best options to provide us with the concepts regarding our research, as well it is one of the best options to implement our work.

2.5 Argumentation

Argumentation can be seen as the principled interaction of different, potentially conflicting arguments, for the sake of arriving at a consistent conclusion [78].

The survey presented in [78] states that argumentation in multi-agent systems has two main lines of research: (i) *autonomous agent reasoning*, such as belief revision and decision-making

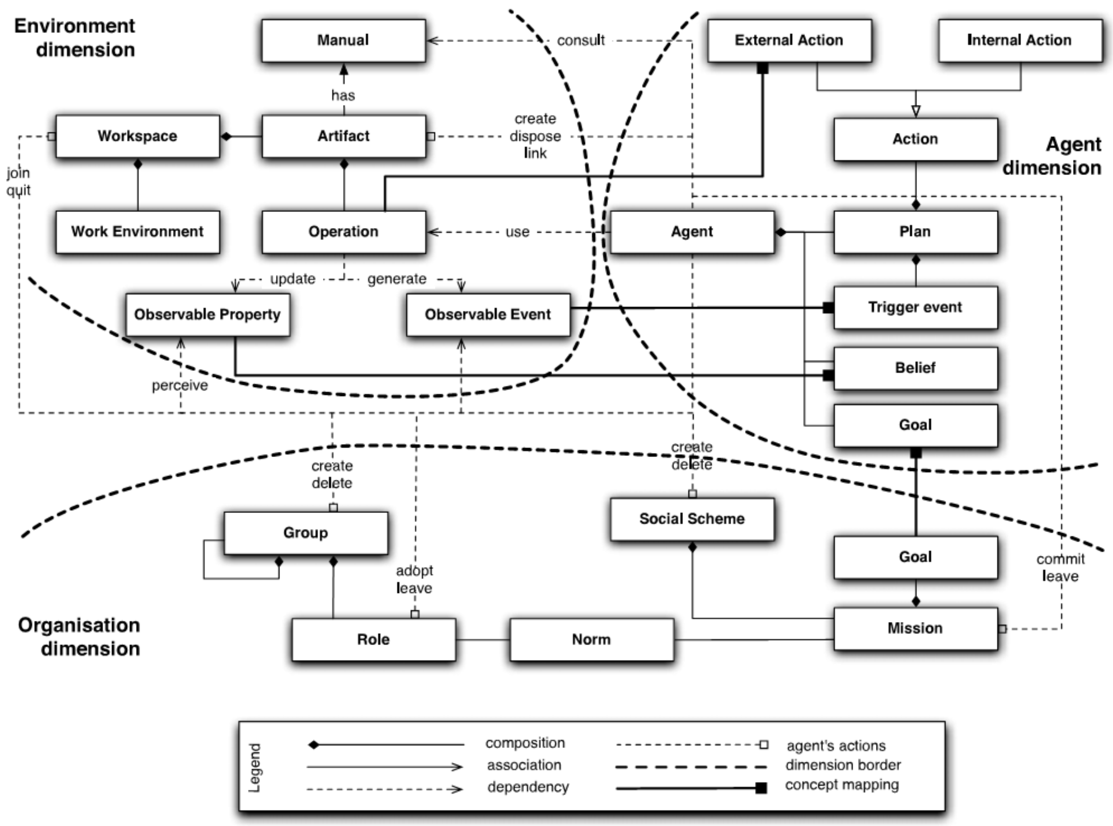


Figure 2.3 – JaCaMo programming Meta-Model [31].

under uncertainty; and (ii) as a vehicle for facilitating *multi-agent interaction*, because argumentation naturally provides tools to designing, implementing and analysing sophisticated forms of interaction among rational agents.

According to [78], argumentation lends itself naturally to two main sorts of problems encountered in multi-agent systems:

- **Forming and revising beliefs and decisions**: Argumentation provides means for forming beliefs and decisions on the basis of incomplete, conflicting, or uncertain information. This is because argumentation provides a systematic means for resolving conflicts among different arguments and arriving at consistent, well-supported standpoints. It also provides a more dialectical procedure for arriving at acceptable conclusions/decisions, given that different pros and cons can be compared for such conclusions/decisions [55];
- **Rational interaction**: Argumentation provides means for structuring dialogues between participants that have potentially conflicting viewpoints. In particular, argumentation provides a framework for ensuring that interaction among agents respects certain principles (*e.g.*, consistency of each participant's statements).

"As a reasoning mechanism, argumentation provides an alternative way to mechanise nonmonotonic reasoning. Argument-based frameworks view the problem of nonmonotonic reasoning as a process in which arguments for and against certain conclusions are constructed and compared. Nonmonotonicity arises from the fact that new premises may enable the construction of new arguments to support new beliefs, or stronger counterarguments against existing beliefs. As the number of premises grows, the set of arguments that can be constructed from those premises grows monotonically. However, because new arguments may overturn existing beliefs, the sets of beliefs may grow nonmonotonically" [78]. Essentially, argumentation can be used both for theoretical reasoning (reasoning about what to believe) as well as practical reasoning (reasoning about what to do) [78, 125].

In the communication/interaction strand, an inherent characteristic of multi-agent systems is that agents need to communicate in order to achieve their individual or collective goals. Agent communication with argumentation techniques allows agents to exchange arguments to justify their stance and to provide reasons that defend their claims. This improved expressivity has many potential benefits, but it is often claimed that it should, in particular [78]:

- make communication more efficient by allowing agents to reveal relevant pieces of information when it is required during a conversation;
- allow for a verifiable semantics based on the agents' ability to justify their claims (and not on private mental states); and
- make protocols more flexible, by replacing traditional protocol-based regulation by more sophisticated mechanics based on commitments.

On the other hand, this improved expressivity comes with a price. According to [78], it poses some serious challenges when it comes to designing autonomous agents that actually communicate by means of argumentation, and makes more difficult:

- the integration with agents' reasoning, which requires to precisely specify what agents should respond to others' agents on the basis of their internal state, but also on the basis of their goals (a strategy for the agent to participate in the interaction);
- the validation of provable desirable properties of these protocols.

One of the most influential work in argumentation is abstract argumentation [45], where the content of individual arguments is not relevant and an overall structure of the relations between arguments is used instead. Abstract argumentation frameworks have their origins in [45], which studies the acceptability of arguments. In [45], the focus is on the attack relation between arguments, and the sets of arguments that defend its members, representing the ones that, given a set of arguments, are acceptable. As most of the work found in the literature make reference to properties and concepts defined in [45], we dedicate the next section to describe abstract argumentation. We describe abstract argumentation for completeness, because some of these definitions and concepts are used in the remainder of the document, even though our approach is not based on absract argumentation frameworks.

2.6 Abstract Argumentation

Dung showed in [45] that argumentation can be studied without consideration to the internal structure of the individual arguments (which became known as abstract argumentation). In his work, arguments are nodes in an argument graph and arcs in this graph represent attack relationships between arguments. Formally:

Definition 1 (Argumentation framework). *An argumentation framework is a pair $AF = \langle A, R \rangle$ where A is a finite set of arguments and $R \subseteq AXA$ is an* attack *relation (also known as* defeat *relation). An argument Arg_1 attacks an argument Arg_2 if $(Arg_1, Arg_2) \in R$.*

A simple example is show in Figure 2.4, where argument *a1* has two attackers (i.e., counterarguments) *a2* and *a4*; furthermore, *a2* is attacked by *a3* and *a4* is attacked by *a5*.

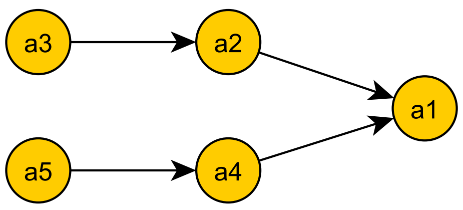


Figure 2.4 – Simple argument graph.

To evaluate an argument consists of checking whether this argument is acceptable or not given the other arguments and the attack relation. The acceptability of arguments is defined by a logical semantics[2], where it is considered how an argument interacts with the other arguments. Below, we have some definitions to study the acceptability of arguments.

Definition 2 (Conflict-free [45]). *Let $\langle A, R \rangle$ be an argument framework and S a set of arguments ($S \subseteq A$). S is conflict-free if no argument in that set attacks another. S defends an argument if it attacks all the attackers of this argument.*

This property, *conflict-free*, is important when thinking of rational agents. Rational agents should have a conflict-free position (the set of arguments to position itself about a particular conclusion) when making a decision or engaging in an argumentation-based dialogue.

For example, in Figure 2.4, arguments $\{a3, a5\}$ defend $a1$. The collective acceptability of a set of arguments can be characterised by various different semantics, as defined below.

Definition 3 (Admissible set [45]). *Let S to be a conflict-free set of arguments in a framework AF. S is admissible if it is conflict-free and defends every element in S.*

[2]Argumentation semantics can be considered "receipts" to evaluate the acceptability of arguments.

An arguments is *admissible* if it is a conflict-free set that defends itself against all attackers. In Figure 2.4, the sets: $\emptyset$, $\{a3\}$, $\{a5\}$, $\{a3, a5\}$ and $\{a1, a3, a5\}$ are all admissible.

Thus, for example, an agent is able to have a consistent position when using any of those admissible set of arguments. For example, it could be irrational for an agent to consider the set $\{a1, a5\}$, given that it knows the existence of the attack between $a2$ and $a1$. Thus, considering the argument graph in Figure 2.4, the only rational way to consider the argument $a1$ in the extension (set of acceptable arguments) is also to consider $a3$ and $a5$ in the extension, given they are necessary to defend $a1$ against $a2$ and $a4$, respectively.

Definition 4 (Complete extension [45]). *An admissible set S is a complete extension if, and only if, all arguments defended by S are also in S (i.e., S is a fixed point).*

In the example in Figure 2.4 the only *complete extension* is the set $\{a1, a3, a5\}$.

The *complete extension* has some refinements, for example, for S a set of arguments:

- S is a *grounded extension* if it is the minimal complete-extension (S is the least fixed point).
- S is a *preferred extension* if it is a maximal complete extension (S is the maximal admissible set).

The *grounded extension* is unique and contains all arguments that are not attacked, as well as the arguments that are defended directly or indirectly by non-attacked arguments.

As for the definition of the acceptability of sets of arguments, we can define the status of individual arguments:

Definition 5 (Argument status [45]). *Let $\langle A, R\rangle$ to be an argumentation system, and $E_1, ..., E_n$ its extensions under a given semantics. For $Arg \in A$ an individual argument, we say that:*

- *Arg is skeptically accepted iff $\forall E_i . Arg \in E_i$, with $i = 1, ..., n$.*
- *Arg is credulously accepted iff $\exists E_i . Arg \in E_i$.*
- *Arg is rejected iff $\nexists E_i . Arg \in E_i$.*

Normally, the literature in argumentation provides structure for arguments when the content of such arguments become necessary, and after defining such argument structure and identifying the attacks relations that come from such content, it is possible to instantiate an abstract argumentation framework [45] in order to evaluate the acceptable arguments. For example, one of the latest instantiations of Dung's abstract formalism appears in Prakken's work [123]. Prakken [123] defines the structure of argument using two types of inference rules, *strict* and *defeasible* rules, respectively. Further, this work defines three types of attack between arguments, *undercutting* and *rebutting* attack (originally formalised in [121]), also the third type called *undermining* attack (inspired by [152]). In this latest instantiation, the author justifies that the structure of arguments permits more expressible representations of the attack relation. In fact, when we are interested in multi-agent programming, it is necessary to define some structure for the arguments, in order to enable them to be interpreted by agents and to give some sort of meaning to them (normally referencing agent attitudes like intentions, beliefs, goals, etc.). Therefore, arguments could be represented in the agent-oriented programming language itself, in order to be (directly) interpreted and understood by agents. Besides Prakken's work [123], which is one of the branches of the ASPIC$^+$ framework [88], in the next section we describe the main approach for structured argumentation, most of them based on abstract argumentation but with a given structure for the arguments, based on different perspectives and logics.

2.7 Structured Argumentation

In abstract argumentation, although the approach provides a clear and precise approach to formalise aspects of argumentation, the arguments are treated as atomic, i.e., the content of such arguments is not formalised, thus all arguments are treated as equal [21]. In order to understand individual arguments, it is necessary to provide content to them, which leads to the idea of "instatianting" abstract argumentation with structured arguments.

There are four main approaches for structured argumentation in the literature, named the ASPIC$^+$ Framework [88], deductive (based on classical logic) arguments [21], Defeasible Logic Programming (DeLP) [54], and Assumption-Based Argumentation (ABA) [144]. In this section we discuss each one of them, in particular the work that gave inspiration and theoretical background for our argumentation framework in agent-oriented programming languages to be described in Section 4.

2.7.1 ASPIC$^+$ Framework

The ASPIC$^+$ Framework has been originated from the European ASPIC Project that ran from 2004 to 2007, which aimed to integrate and consolidate the main approaches to structured argumentation. Such project has relevant publications, starting from [35], which introduced the desirable properties for structured argumentation frameworks (called *rationality postulates*), [123, 87], some instantiations for concrete logics, e.g., [20], and other structured general accounts of argumentation [6]. The last main publication related to ASPIC$^+$ Framework is [88], and we describe such structured argumentation framework based on that last work, which is strongly based on the previous work [87].

ASPIC$^+$ was designed to include *strict* and *defeasible* inferences. The first kind is arguments constructed from deductive or *strict* inference rules, which draw conclusions from premises using deductive inferences. However, in order to cover the fallibility of an argument, which is not only present in their premises, arguments in ASPIC$^+$ can be constructed using *defeasible* inference rules. Such *defeasible* arguments can be attacked on the application of such defeasible inference rules, given that such defeasible rules are interpreted as premises presumptively, rather than deductively, supporting their conclusions.

Therefore, the ASPIC+ framework is described as a triple $\langle \mathcal{L}, \mathcal{R}, n \rangle$, where:

- $\mathcal{L}$ is a logical language closed under negation ($\neg$).
- $\mathcal{R} = \mathcal{R}_s \cup \mathcal{R}_d$ is a set of *strict* ($\mathcal{R}_s$) and *defeasible* ($\mathcal{R}_d$) inference rules of the form $\varphi_1, \ldots, \varphi_n \rightarrow \varphi$ and $\varphi_1, \ldots, \varphi_n \Rightarrow \varphi$ respectively.
- n a partial function $n : \mathcal{R}_d \rightarrow \mathcal{L}$.

The inference rules in ASPIC+ framework are not object-level formulae in the language $\mathcal{L}$, but are meta to the language [88]. Then, when using the ASPIC+ framework, it is necessary to specify the so called knowledge base $\mathcal{K}$, containing the premises (separated into ***axioms*** $\mathcal{K}_n$ and ***ordinary premises*** $\mathcal{K}_p$). Together, the argumentation system and the knowledge base are an argumentation theory.

To explain what an argument is, [88] introduces the following functions: `Prem` which returns the formulæ of $\mathcal{K}$ used to build the argument, `Conc` which returns the conclusion, `Sub` which returns the sub-arguments, `DefRules` which returns all the defeasible rules of the argument, and `TopRule` which returns the last inference rule used in the argument. Therefore, an argument Arg_i on the basis of an argumentation theory with the knowledge base $\mathcal{K}$ and an argumentation system $\langle \mathcal{L}, \mathcal{R}, n \rangle$ is:

1. φ if $\varphi \in \mathcal{K}$ with $\texttt{Prem}(\texttt{Arg}_\texttt{i}) = \{\varphi\}$, $\texttt{Con}(\texttt{Arg}_\texttt{i}) = \{\varphi\}$, $\texttt{Sub}(\texttt{Arg}_\texttt{i}) = \{\varphi\}$, $\texttt{DefRules}(\texttt{Arg}_\texttt{i}) = \emptyset$, $\texttt{TopRule}(\texttt{Arg}_\texttt{i})$ = undefined.

2. $Arg_1, \ldots, Arg_n \rightarrow \psi$ if $Arg_1, \ldots, Arg_n$ are arguments such that there exists a strict rule $\texttt{Conc}(\texttt{Arg}_1), \ldots, \texttt{Conc}(\texttt{Arg}_\texttt{n}) \rightarrow \psi$ in $\mathcal{R}_s$.
   $\texttt{Prem}(\texttt{Arg}_\texttt{i}) = \texttt{Prem}(\texttt{Arg}_1) \cup \ldots \cup \texttt{Prem}(\texttt{Arg}_\texttt{n})$
   $\texttt{Conc}(\texttt{Arg}_\texttt{i}) = \psi$
   $\texttt{Sub}(\texttt{Arg}_\texttt{i}) = \texttt{Sub}(\texttt{Arg}_1) \cup \ldots \cup \texttt{Sub}(\texttt{Arg}_\texttt{n}) \cup \{\texttt{Arg}_\texttt{i}\}$
   $\texttt{DefRules}(\texttt{Arg}_\texttt{i}) = \texttt{DefRules}(\texttt{Arg}_1) \cup \ldots \cup \texttt{DefRules}(\texttt{Arg}_\texttt{n})$
   $\texttt{TopRule}(\texttt{Arg}_\texttt{i}) = \texttt{Conc}(\texttt{Arg}_1), \ldots, \texttt{Conc}(\texttt{Arg}_\texttt{n}) \rightarrow \psi$

3. $Arg_1, \ldots, Arg_n \Rightarrow \psi$ if $Arg_1, \ldots, Arg_n$ are arguments such that there exists a defeasible rule $\texttt{Conc}(\texttt{Arg}_1), \ldots, \texttt{Conc}(\texttt{Arg}_\texttt{n}) \Rightarrow \psi$ in $\mathcal{R}_d$.
   $\texttt{Prem}(\texttt{Arg}_\texttt{i}) = \texttt{Prem}(\texttt{Arg}_1) \cup \ldots \cup \texttt{Prem}(\texttt{Arg}_\texttt{n})$
   $\texttt{Conc}(\texttt{Arg}_\texttt{i}) = \psi$
   $\texttt{Sub}(\texttt{Arg}_\texttt{i}) = \texttt{Sub}(\texttt{Arg}_1) \cup \ldots \cup \texttt{Sub}(\texttt{Arg}_\texttt{n}) \cup \{A\}$
   $\texttt{DefRules}(\texttt{Arg}_\texttt{i}) = \texttt{DefRules}(\texttt{Arg}_1) \cup \ldots \cup \texttt{DefRules}(\texttt{Arg}_\texttt{n}) \cup \{\texttt{Conc}(\texttt{Arg}_1), \ldots, \texttt{Conc}(\texttt{Arg}_\texttt{n}) \Rightarrow \psi\}$
   $\texttt{TopRule}(\texttt{Arg}_\texttt{i}) = \texttt{Conc}(\texttt{Arg}_1), \ldots, \texttt{Conc}(\texttt{Arg}_\texttt{n}) \Rightarrow \psi$

An argument Arg_i is called a *strict argument* when $\texttt{DefRules(Arg}_\texttt{i}\texttt{)} = \emptyset$; a *defeasible argument* when $\texttt{DefRules(Arg}_\texttt{i}\texttt{)} \neq \emptyset$; *firm* when $\texttt{Prem(Arg}_\texttt{i}\texttt{)} \subseteq \mathcal{K}_n$; and *plausible* when $\texttt{Prem(Arg}_\texttt{i}\texttt{)} \cap \mathcal{K}_p \neq \emptyset$.

Arguments in the ASPIC⁺ Framework have three kinds of attacks[3]:

- *Undercut*: an argument Arg_1 *undercuts* an argument Arg_2 (on Arg_i) iff $\texttt{Conc(Arg}_1\texttt{)} = \overline{r}$, where r is the top rule of Arg_i, for some $Arg_i \in \texttt{Sub(Arg}_2\texttt{)}$.
- *Rebut*: an argument Arg_1 *rebuts* an argument Arg_2 (on Arg_i) iff $\texttt{Conc(Arg}_1\texttt{)} = \overline{\varphi}$ for some $Arg_i \in \texttt{Sub(Arg}_2\texttt{)}$ of the form $Arg_j, \ldots, Arg_k \Rightarrow \varphi$.
- *Undemine*: an argument Arg_1 *undermines* an argument Arg_2 (on φ) iff $\texttt{Conc(Arg}_1\texttt{)} = \overline{\varphi}$ for an ordinary premise φ of Arg_2.

Considering the attack relation and preferences between arguments, in ASPIC⁺, the defeat relation is defined as follows::

- Arg_1 successfully rebuts Arg_2 if Arg_1 rebuts Arg_2 on Arg_i and Arg_1 is not less preferred than Arg_i.
- Arg_1 successfully undermines Arg_2 if Arg_1 undermines Arg_2 on φ and Arg_1 is not less preferred than φ.
- Arg_1 defeats Arg_2 iff Arg_1 undercuts or successfully rebuts or successfully undermines Arg_2.

With the defeat relation, the acceptability of arguments in ASPIC⁺ can be defined using Dung-style abstract argumentation frameworks [45], as described by [88].

For ASPIC⁺ [88], the strict rules, axioms and defeasible rules are domain specific knowledge, which need to be defined for a particular domain based on the logical language used in ASPIC⁺.

2.7.2 Deductive Argumentation

Deductive argumentation has been proposed in [21], where the authors threat arguments as deductive instances of abstract arguments from Dung's work [45]. In deductive argumentation, a conclusion is derived from premises using one or more inference steps, which are infallible, in the sense that there is no uncertainty in such inferences. This means that when we accept the premises, we should accept the intermediate conclusions of each inference step and the final conclusion as well [21]. In [21], the authors assume that deductive reasoning is formalised by a monotonic logic, and each deductive argument is a pair formed by a set of premises that logically

[3]We use a general operator for contradictory information, where $\overline{\varphi}$ is contradictory to φ, that is, $\overline{\varphi} \equiv \neg\varphi$, $\overline{\neg\varphi} \equiv \varphi$, and so on.

entails a conclusion, the first and second item respectively, i.e., an argument $\langle S, c \rangle$, where $S \vdash c$ ($\vdash$ represents the classical consequence relation). The authors [21] argue that the benefits of deductive argumentation include: (i) the explicit representation of the information used to support the claim (conclusion of that argument); (ii) the explicit representation of the claim; and (iii) a simple and precise connection between the support and the claim by means of consequence relation.

The language used in [21] is based on classical logic (propositional and first-order classical logic), representing argument of the type $p_i, \ldots, p_j \rightarrow p_k$ where $p_i, \ldots, p_j, p_k$ are literals, and modus ponens is the only proof rule. The authors use argument graphs to represent arguments and attacks. In argument graphs, the input is a knowledge base (i.e., a set of logical formula), and the goal is to construct arguments and counterarguments which can be generated from the knowledge base, thus the output is the argument graph.

Regarding arguments constructed in deductive argumentation [21], there are two main constraints in the argument structure. The first is the so called **consistency constraint**, which describe that the support of an argument needs to be consistent, given that inconsistent premises in classical logic are normally useless, considering that we are able to draw any predicate from a contradiction. The second constraint is called **minimality constraint**, which describes that there is no $S' \subset S$ such that $S' \vdash c$. Such property is desired for deductive argumentation because it eliminates irrelevant premises.

In deductive argumentation, a counterargument is an argument that attacks another argument, which is defined in terms of the logical contradiction between the claim of the counter-argument and one premise of the claim of the attacked argument.

In deductive argumentation proposed by [21], considering two arguments Arg_1 and Arg_2, there are two kind of attack:

- *Undercut*: Arg_1 undercuts Arg_2 if there is a simple rule $p_i, \ldots, p_n \rightarrow p_j$ in `Support`(Arg_2) and there is a $p_k \in \{p_i, \ldots, p_n\}$ such that `Claim`(Arg_1) is the complement of p_k.
- *Rebut*: Arg_1 rebuts Arg_2 if `Claim`(Arg_1) is the complement of `Claim`(Arg_2).

Based on the attack relation, the arguments can be structured in argument graphs, given the attack relation between them. Looking for argument graphs, we are able to identify the acceptable arguments, similar to Dung's work [45].

2.7.3 Defeasible Logic Programming (DeLP)

Defeasible Logic Programming [53, 54] provides a computational reasoning system that uses an argumentation engine to obtain answers from a knowledge base represented using a logic programming language extended with defeasible rules. Defeasible logic programming can be seen as a formalisation of defeasible reasoning in which the results of logic programming and argumentation are combined [54]. The language in Defeasible Logic Program (DeLP for short) contains classical negation and offers the possibility of using default negation [54], including the capability for representing knowledge declaratively, and addicting the possibility of representing *defeasible rules*, i.e., a weak kind of inference.

The authors argue that, from the research line in nonmonotonic and defeasible reasoning, logic programming has emerged as one of the more attractive choices for its theoretical soundness and effective implementations. Also, they describe that using defeasible reasoning and logic programming is a natural choice to address the problem of inconsistency [54].

In DeLP, the argumentation-based inference mechanism is able to consider reasons for and against conclusions, deciding which ones can be obtained (warranted) from the knowledge base [53, 54]. A defeasible logic program is a set of facts and rules, i.e., (Π, Δ) with Π the set of facts and strict rules and Δ the set of defeasible rules.

Arguments in DeLP are pairs $\langle \mathcal{A}, \mathcal{L} \rangle$, with $\mathcal{L}$ a ground literal and $\mathcal{A}$ the argument for $\mathcal{L}$, $\mathcal{A}$ is a minimal set of defeasible rules - $\mathcal{A} \in \Delta$ - such that (i) there exists a defeasible derivation for $\mathcal{L}$ from $\Pi \cup \mathcal{A}$; (ii) no pair of contradictory literals can be defeasibly derived from $\Pi \cup \mathcal{A}$; and (iii) if $\mathcal{A}$ contains some rule with an extended literal ***not F***, then the literal F cannot be in the defeasible derivation of $\mathcal{L}$ from $\Pi \cup \mathcal{A}$ [54].

An argument $\langle \mathcal{B}, \mathcal{Q} \rangle$ is an counterargument for $\langle \mathcal{A}, \mathcal{L} \rangle$ at literal $\mathcal{P}$, if there exists a subargument $\langle \mathcal{C}, \mathcal{P} \rangle$ of $\langle \mathcal{A}, \mathcal{L} \rangle$, i.e., $\mathcal{C} \subseteq \mathcal{A}$, such that $\mathcal{P}$ and $\mathcal{Q}$ disagree (they are contradictory). When $\langle \mathcal{B}, \mathcal{Q} \rangle$ is an counterargument for $\langle \mathcal{A}, \mathcal{L} \rangle$, the authors describe that $\langle \mathcal{B}, \mathcal{Q} \rangle$ *attacks* $\langle \mathcal{A}, \mathcal{L} \rangle$, and that $\langle \mathcal{B}, \mathcal{Q} \rangle$ and $\langle \mathcal{A}, \mathcal{L} \rangle$ are in *conflict* [54].

DeLP works by means of queries. When we want to know if a literal $\mathcal{Q}$ is warranted from (Π, Δ), we query $\mathcal{Q}$ in Defeasible Logic Programming. There are four possible answers for a query $\mathcal{Q}$ posed to a DeLP-program $\mathcal{P}$:

- `YES`, if $\mathcal{Q}$ is warranted from $\mathcal{P}$;
- `NO`, if the complement of $\mathcal{Q}$ is warranted from $\mathcal{P}$;
- `UNDECIDED`, if nor $\mathcal{Q}$ nor its complement is warranted from $\mathcal{P}$; and
- `UNKNOWN`, if $\mathcal{Q}$ is not a signature of the program $\mathcal{P}$.

Informally, a literal $\mathcal{Q}$ will be warranted if there exists at least an argument $\mathcal{A}$ supporting $\mathcal{Q}$ that prevails after going through a dialectical process considering all its counterarguments [54].

DeLP also allows us to define preferences between arguments, and the authors [54] argue that such preferences are modular in the argumentation system, and they could be defined by different criterion established over the set of arguments. However, based on [53], the authors describe that an argument is preferred to another if there is at least one rule in the first argument which is preferred to the second.

The acceptability of an argument in DeLP is given by a *dialectical tree*, where the root node is the argument for the queried literal. Each child is a defeater of subarguments in the root node, and their child defeaters of them, and so on. In other words, a literal is warranted if there exists an argument for such literal, and all the defeaters for such argument are defeated by other arguments.

One of the last proposal for DeLP is the so called *DeLP Servers* [52, 54]. In that work, the authors propose that reasoning can be seen as a service that can be offered as part of a knowledge base infrastructure, then client agents distributed in remote hosts are able to consult different reasoning services implemented as DeLP-Servers that can be also distributed. In such an approach, both common or public knowledge can be stored in a server and represented as a DeLP-program, and it works with the client sending a query to DeLP-Server and receiving the corresponding answer. The DeLP-Server uses the knowledge stored in it to answer a query, but offering the the possibility to integrate the agent "private" knowledge, which can be sent as part of the query. Agents can execute queries in parallel and only their private knowledge will be used in their queries. Therefore, as described by the authors [54], a client cannot make permanent changes to the public DeLP-program stored in a server.

Furthermore, the authors, in [54], describe that DeLP could implement explanations for answers, which is an important topic in several areas of Artificial Intelligence. In particular, the approach presented by them aims to explain the understanding of how the warranted status of a particular argument was obtained from a given argumentation framework.

2.7.4 Assumption-Based Argumentation (ABA)

Assumption-based Argumentation (ABA for short) [46, 144] was developed starting in the 90s, aiming to be a computational framework to reconcile and generalise most existing approaches to default reasoning. ABA take inspiration from the Dung's preferred extension semantics for logic programming [44], considering its dialectical interpretation for acceptability with negation-as-failure assumption-based on the notion of "no-evidence-to-the-contrary" [44], and Dung's abstract argumentation [45].

ABA is an instance from Dung's abstract argumentation for determining the *acceptability* of arguments. Therefore, all semantic notions in abstract argumentation apply to ABA as well [46, 144]. While, in abstract argumentation, the arguments are abstract and primitive, in ABA arguments are deductions using inference rules supported by *assumptions*. Attacks between two arguments are deductions of one argument for the contrary of an assumption in the support of the second argument. Differently from deductive argumentation presented by [12] and DeLP presented by [54], ABA does not have rebuttals, and does not impose that arguments need to have consistent and minimal supports. ABA uses the notion of *relevant* and *largely consistent* arguments [46].

The acceptability of arguments is given by a computational machinery in the form of ***dispute derivation***, where it is constructed considering a dialectical structure of the proponent's argument for a claim, an opponent's counterargument attacking the argument, the proponent's arguments attacking all opponent's counterarguments, and so on, similar to the so called ***dialectical tree*** in DeLP. As such an approach has its root in logic programming, it has the advantage of a fine level of granularity, given it offers an interleaving of construction of argument and determining their acceptability [46]. Arguments are represented by tree-structures, such that the argument generation can be performed by means of a proof procedure, which searches the space of applications of inference rules [46].

Attacks in ABA are defined in terms of ***contrary*** of assumptions. Considering two arguments, denoted by $S_1 \vdash c_1$ and $S_2 \vdash c_2$ (with S_1 and S_2 the set of assumptions that support the conclusions c_1 and c_2, respectively), the first argument attacks[4] the second if c_1 is the contrary of an assumption in S_2 [46]. In order to determine the acceptability of a claim in ABA, agents need to find an argument for the claim that can be defended against attacks from other arguments. The process to define the acceptable arguments in ABA is similar to abstract argumentation, thus [46] describes that: (i) a set of arguments is ***admissible*** iff it does not attack itself and it attacks every argument that attacks it; (ii) an admissible set of arguments is ***complete*** if it contains all arguments that it ***defends***; and (iii) the least complete set of arguments is ***grounded***.

In all frameworks, or also called argumentation systems, for structured argumentation presented in this section, there is a consensus that domain-dependent knowledge must be specified according to the application to be developed, in order to generate and evaluate arguments for that particular domain. In other words, an argumentation framework is used to generate arguments and evaluate the acceptability of such arguments, but which arguments the framework will be able to generate and evaluate depends on the knowledge available in that application domain, and domain-dependent reasoning patterns. In the argumentation field, there are seminal studies in reasoning patterns used for argumentation, which come from a more philosophical point of view, and they have great potential for the field of artificial intelligence, particularly for applications in multi-agent systems (although they have been largely unexplored), the so called ***argumentation schemes*** [156]. Therefore, we propose to use argumentation schemes as a meta-level of reasoning patterns that agents could use to instantiate arguments depending on the knowledge available to them. To this end, we introduce argumentation schemes in the next section.

2.8 Argumentation Schemes

Argumentation schemes are considered deductive and inductive forms of argument, added the so called ***defeasible***, ***presumption*** or ***abductive*** part. Such an argument, considered ***defeasible***, may not be very strong by itself, but may be strong enough to provide evidence to warrant

[4]It is important to note that attacks between arguments depends only on attacking assumptions (***undercutting***).

rational acceptance of its conclusion, given that its premises are acceptable [149] (the argumentation schemes, also, are considered attempts to conclusions under conditions of uncertainty and lack of knowledge).

The conclusion of a defeasible argument can be accepted tentatively in relation to the evidence known, but may need to be retracted as new evidence come in. In [156] is argued that the most important kinds of schemes are defeasible in nature, meaning that even after the argument has been accepted, it might later be defeated as new evidence enter into consideration. The factor of defeasibility raises the problem of how schemes are rationally binding. The critical questions that are possible turn out problematic conclusions in several aspects.

The defeasibility generally is linked with a dialogue, where a proponent, based in a scheme, assert some conclusion and the opponent, also based in the scheme, may make a critical question which needs to be successfully answered by the proponent [156].

"Scheme hold great potential for tackling a variety of problems in artificial intelligence. The real world represents an immerse challenge to artificial agents. Even if we focus only upon reasoning capabilities, and leave to one side the physical aspects of interacting with the world, an agent must deal with two fundamental problems: uncertainty and incompleteness" [156]. Typically such reasoning systems will have to interact not only with the world, but also with humans, which need to be understood dialectically.

Together, argumentation schemes and critical questions, are used to evaluate a given argument in a particular case, in relation to a context of dialogue in which the argument occurred. In an argumentation scheme if all premises are supported by some weight of evidence, then the weight of acceptability is shifted toward the conclusion, subject to rebuttal by the asking of appropriate critical questions. To judge the strength of an argument is used critical questions, and critical questions are a benefit of the use of scheme-based approach [156].

To exemplify our approach, we adapted the argumentation schemes *Argument from Position to Know* from [153] to a multi-agent (organisational) platform, so that for example roles that agents play in the system can be referred to within the scheme. Consider the *Argument from role to know in multi-agent systems* (*role to know* for short) :

> "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?

The critical questions are to point doubt on the structural link between the premise and the conclusion. To judge the strength or weakness of an argument based on a scheme is used associated critical questions (this allow us to judge if the argument is good or fallacious). Critical questions are vital in the definition of a scheme. The options, in a dialogue, to an opponent after receiving an argument are: (i) ask a critical question related to that argument; (ii) provide an argument against the claim of the argument received; (iii) challenge one of the premises of that argument; or (iv) accept the conclusion of that argument as a commitment.

The connection between argumentation schemes and argumentation frameworks is not clear in the literature of argumentation. In particular, how to represent argumentation schemes in structured argumentation frameworks also is an open question. Although some authors, for example [123], have suggested that argumentation schemes could be represented as defeasible inference rules and the acceptability of argument evaluated in frameworks as ASPIC+ [88], the role of the critical questions seems not fully approached. In Section 3 we propose a formal and practical representation for argumentation schemes in multi-agent systems, as well as we formally define the role of the critical questions in the process of defining the acceptability of arguments.

3. ARGUMENTATION SCHEMES IN MULTI-AGENT SYSTEMS

As described in Section 2.8, argumentation schemes are well-known structures used to model reasoning patterns, which agents will use to instantiate arguments. Different applications domains require different argumentation schemes, and some argumentation schemes are domain dependent and they could be specified for a particular application in multi-agent systems context. Examples of argumentation schemes defined for particular domains are found in: [145] where they are used for analysing the provenance of information, [113] where they are used by agents for reasoning about trust, [142] where they are used by agents to argue about human organs transplantation, [58] where they are used to guide the process of data mining in the biomedical research domain, [72] where they are used to support different treatment options to patients, our own work presented in [100] where we define two argumentation schemes to implement argumentation-based interface agents for data access control between smart applications, and so forth.

Besides argumentation schemes are domain-dependent reasoning patterns for argumentation, as the literature suggests, they share a common structure of elements. Argumentation schemes are represented by a set of *premises*, a *conclusion* that can be drawn from the premises, and a set of *critical questions* that point out possible problems on using that scheme [156]. Thus, argumentation schemes can be formalised as:

Definition 6 (Argumentation Scheme). *Formally, an argumentation scheme is a tuple $\langle \mathcal{SN}, \mathcal{C}, \mathcal{P}, \mathcal{CQ} \rangle$ with $\mathcal{SN}$ the argumentation scheme identifier (name), $\mathcal{C}$ the conclusion of the argumentation scheme, $\mathcal{P}$ the premises, and $\mathcal{CQ}$ the associated critical questions.*

Although argumentation schemes have a well defined general structure, they have some particularities that make them difficult to model in computational languages, mainly because they represent stereotypical reasoning patterns that are considered defeasible [156]. Some approaches in the argumentation literature have suggested that argumentation schemes [156] could be translated into defeasible inferences [123, 103, 102], and the acceptability of arguments, instantiated using these rules, could be checked using frameworks such as ASPIC+ [88], DeLP [54], and others [101, 109]. However, for this work, the role of the critical questions seems to be missing. That occurs because critical questions might point out doubts not only about the premises and inference rules used in an argument (which could be easily verified in those frameworks) but also about presumptions used in that reasoning pattern, which are not explicitly present in the argument [156] and, consequently, they cannot be verified in those frameworks.

In order to make explicit the representation of arguments, we introduce the language $\mathcal{L}$ used in this document. We use a first-order language as the basis for our representation, given that most agent-oriented programming languages are based on logic programming, including Jason [33], and we are interested not only on the formal specification of our framework but also on its implementation. In particular, we have a set of atomic formulæ $\{p_0, \dots, p_n\} \in \mathcal{L}$, and a set of defeasible inferences rules $\{(p_i, \dots, p_j \Rightarrow p_k), \dots, (p_l, \dots, p_m \Rightarrow p_o)\} \in \mathcal{L}$. In order to refer to the unification process, we use a most general unifier θ (as usual in logic programming). We use uppercase letters to represent variables — e.g., `Ag` and `R` in `role(Ag,R)` — and lowercase letters to represent terms and ground literals — e.g., `john`, `doctor` and `role(john,doctor)`. We use "$\neg$" to represent strong negation in $\mathcal{L}$, e.g., $\neg$`honest(pietro)` means that `pietro` is ***not*** honest. We also use ***negation as failure*** "`not`" to represent the absence of information, e.g., `not(honest(pietro))` means that an agent is ignorant about (does not know) if `pietro` is honest[1]. Atomic formulæ, set of atomic formulæ, and defeasible inference rules can be annotated with relevant information used in the inference mechanism[2]. In this document, annotations are grounded literals representing the names of argumentation schemes, e.g., $p_{l_{[sn]}}$ and $\{p_i \wedge \ldots \wedge p_j\}_{[sn]}$ are used to represent critical questions related to the argumentation scheme sn, and $(p_l, \ldots, p_m \Rightarrow p_o)_{[sn]}$ is used to represent the defeasible inference rule corresponding the inference modelled by the argumentation scheme sn.

For example, the argumentation scheme *role to know*, introduced in Section 2.8, can be represented using the following defeasible inference rule:

$(\texttt{asserts(Agent,Conclusion),role(Agent,Role),role\_to\_know(Role,Domain),}$
$\texttt{about(Conclusion,Domain)} \Rightarrow \texttt{Conclusion})_{\texttt{[as(role\_to\_know)]}}.$

with the argumentation scheme name $\mathcal{SN}$ = `role_to_know`, the conclusion $\mathcal{C}$ = `Conclusion`, the premises $\mathcal{P}$ = $\{$`role(Agent,Role)`, `role_to_know(Role,Domain)`, `asserts(Agent,Conclusion)`, `about(Conclusion,Domain)`$\}$. The associate critical questions $\mathcal{CQ}$ are:

- $\texttt{role\_to\_know(Role,Domain)}_{\texttt{[as(role\_to\_know)]}}.$
- $\texttt{honest(Agent)}_{\texttt{[as(role\_to\_know)]}}.$
- $\texttt{asserts(Agent,Conclusion)}_{\texttt{[as(role\_to\_know)]}}$
- $\texttt{role(Agent,Role)}_{\texttt{[as(role\_to\_know)]}}.$

Note that, the second critical question, for example, cannot be identified in the premises of an argument instantiated from this reasoning pattern, thus an agent is able to question that, when reasoning or participating in a dialogue, only when it is aware of the argumentation scheme *role to know*. That means, the agent is not able to find that information in $\mathcal{P}$ (the premises), $\mathcal{C}$ (the conclusion) or even in the inference rule when it is not aware of the scheme, i.e., it is not able to verify that information looking for the representation of arguments in structured argumentation frameworks. One possible solution for that issue, considering the approaches mentioned [123, 103, 102], could be to overload the computational representation of such arguments in order to make explicit

[1]Note that negation as failure is used in logic tests, e.g., in inference rules and context of plans in Jason.

[2]These annotations and the inference mechanism we developed using such annotations are inspired by Labelled Deductive Systems (LDS) [51].

the critical questions related to the scheme used to instantiate that argument, i.e., it is necessary to include an explicit representation of such critical questions as premises into the argument. However, that infringes the requirement of many argumentation frameworks in which the support of an argument should be minimal, e.g., [117, 115]. Otherwise, when such critical questions are not represented in the argument, such information is lost in those approaches. Another option is to make all agents aware of such reasoning pattern (including the associated critical questions). Thus, agents are able to identify from which argumentation scheme an argument has been instantiated, as well as to use the associated critical questions when evaluating such argument. Considering the current directions on the development of multi-agent systems inspired by the concept of *open* systems [66], we argue that the second solution is better than the first one, and the current representation of arguments in multi-agent systems requires argumentation schemes to be shared by all agents in a multi-agent system.

We are considering two alternative ways to represent shared argumentation schemes in multi-agent systems. The first would be specifying argumentation schemes during the conception of a multi-agent system (in their organisational specification). Thus, we have proposed an infrastructure extending the MOISE organisational model [65] with a new dimension for argumentation schemes. In this approach, when the multi-agent system organisation is specified, we are able to specify the reasoning patterns that agents will be able to use in such a system. This approach has been published in [102], it is named Soc$^{\texttt{ARG}}$ model, and it will be presented in Section 3.1. The second approach is based on our previous collaborative work [50, 49, 136], in which argumentation schemes could be specified in domain-dependent semantics data-bases (ontologies) and shared by all agents in a multi-agent system. This last approach will be presented in Section 3.2.

3.1 The Soc$^{\texttt{ARG}}$ model

In this section, we present a formal specification for multi-agent systems called Soc$^{\texttt{ARG}}$, including a new dimension which allows the specification of argumentation schemes that can be used during interaction/reasoning within an instantiated multi-agent systems. As part of our model, we also extend usual normative specifications in order to specify constraints over the use of argumentation schemes (and their instantiated arguments) in the system, considering the roles and context of agents. Further, we describe a series of relations between the types of specifications in our approach.

Formally, Soc$^{\texttt{ARG}}$ is a tuple $\langle \mathcal{SS}, \mathcal{FS}, \mathcal{AS}, \mathcal{NS} \rangle$, with $\mathcal{SS}$ the structural specification, $\mathcal{FS}$ the functional specification, $\mathcal{AS}$ the argumentation-scheme specification, and $\mathcal{NS}$ the normative specification. The *structural specification* ($\mathcal{SS}$) is a tuple $\langle \mathcal{R}, \sqsubset, \mathit{rg} \rangle$ with $\mathcal{R}$ a set of roles, $\sqsubset$ the inheritance relation between roles, including communication link ($\mathit{link}_{\mathit{com}}$), authority link ($\mathit{link}_{\mathit{aut}}$), and acquaintance link ($\mathit{link}_{\mathit{acq}}$), and rg the organisation root group specification. The *functional specification* ($\mathcal{FS}$) is a tuple $\langle \mathcal{M}, \mathcal{G}, \mathcal{S} \rangle$ with $\mathcal{M}$ the set of *missions*, consisting of groupings of collective or individual goals, $\mathcal{G}$ is the set of the *collective* or *individual goals* to be satisfied, and

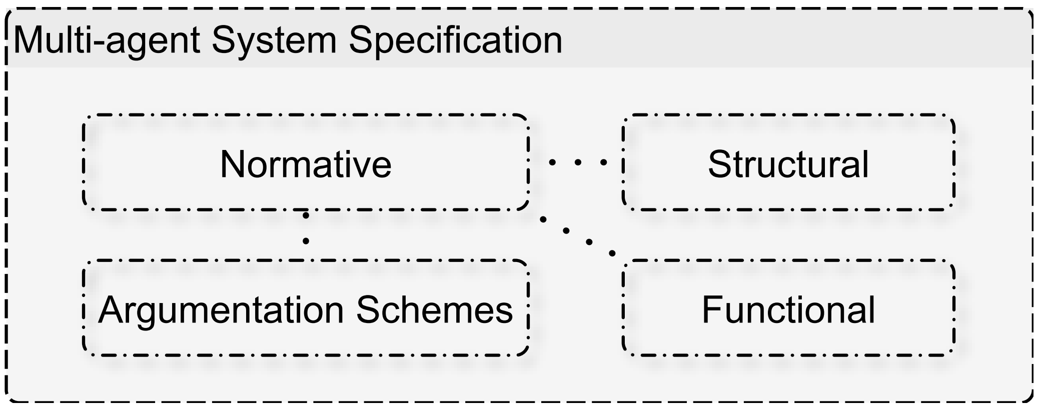


Figure 3.1 – Soc$^{\text{ARG}}$ links between specifications.

$\mathcal{S}$ is the set of ***social schemes***, tree-like structures of plans for goals. The argumentation-scheme specification ($\mathcal{AS}$) is a set of argumentation schemes, each one is a tuple $\langle \mathcal{SN}, \mathcal{C}, \mathcal{P}, \mathcal{CQ} \rangle$ with $\mathcal{SN}$ the argumentation scheme name (which must be unique in the system), $\mathcal{C}$ the conclusion of the argumentation scheme, $\mathcal{P}$ the premises, and $\mathcal{CQ}$ the associated critical questions. The ***normative specification*** ($\mathcal{NS}$) is a set of tuples $\langle id, dm, r, scope \rangle$ with id a norm identifier, dm a normative modality (obligation or permission), r is the role concerned by the normative modality, and $scope$ the norm's scope. This formalisation is inspired in the MOISE organisational model. However, note that our proposal is not tied to the organisational specifications of MOISE, any specification of (open) multi-agent systems using those same concepts (groups, roles, social plans, sets of goals allocated to agents, and simple norms) can be used to combined with our argumentation-scheme specifications for open multi-agent systems.

The specification of argumentation schemes in Soc$^{\text{ARG}}$ is independent of the other organisational specifications, but it is particularly connected to the normative specification which links the structural, functional, and argumentation-scheme specifications. In the argumentation-scheme specification, the argumentation schemes and their corresponding critical questions are defined. After that, in the normative specification, we can specify which argumentation schemes and corresponding critical questions can be used by agents depending on the roles they play in the multi-agent system and the communication links between such roles, both declared in the structural specification. The usage of argumentation schemes and critical questions can also consider the context of the social goals, which are associated with the functional specification. The links between the specifications present in Soc$^{\text{ARG}}$ are represented in Figure 3.1.

In Soc$^{\text{ARG}}$, argumentation schemes could be defined in XML on the top of the organisational specification, but other languages such as AML (Argument Markup Language) introduced in [129], AIF (Argument Interchange Format) [36], could be used instead. This is possible because the high-level language is interpreted by a form of "management infrastructure", like the one developed in [64]. The management infrastructure allows agents developed in different agent-oriented programming languages to participate in the open system and become aware of such specifications (argumentation schemes, norms, goals, etc.) through this interface.

For example, the argumentation scheme *Argument from role to know in multi-agent systems*, is specified in the XML file that specifies the multi-agent systems in Soc$^{\text{ARG}}$. The code in XML is presented below.

```
<argumentation_scheme_specification>
 <argumentation_scheme id="as1" name="role_to_know">
  <conclusion language="Prolog" content="Conclusion"/>
  <premise language="Prolog" content="role(Agent,Role)"/>
  <premise language="Prolog" content="role_to_know(Role,Domain)"/>
  <premise language="Prolog" content="asserts(Agent,Conclusion)"/>
  <premise language="Prolog" content="about(Conclusion,Domain)"/>
  <critical_questions>
   <critical_question id="cq1" content="role_to_know(Role,Domain)"/>
   <critical_question id="cq2" content="honest(Agent)"/>
   <critical_question id="cq3" content="asserts(Agent,Conclusion)"/>
   <critical_question id="cq4" content="role(Agent,Role)"/>
  </critical_questions>
 </argumentation_scheme>
</argumentation_scheme_specification>
```

In order to make explicit the use of argumentation scheme in the normative specification, the MOISE normative specification was extended[3] to the form of $\langle id, dm, r, scope\rangle$, where the $scope$ can assume two forms: (i) $do(m)$ considering the execution of a mission m (the usual case in MOISE model), and (ii) $use(as, cons)$ referring the use of an argumentation scheme as and its respective constraints $cons$. The normative specification in XML is extended as well, to include the `scope` declaration, which allows us to determine if the norm refers to a mission or to the use of an argumentation scheme. When doing so one can specify whether that use is permitted or not, including the constraint related to the critical questions and context. An example is presented in the XML code below.

```
<normative-specification>
<norm id="n1" type="obligation" role="r1">
    <scope type="do" mission="m1"/>
</norm>
<norm id="n2" type="permission" role="r1">
    <scope type="use" arg_scheme="as1">
    <context m_id="m1">
    <except cq_id="cq1" content="role_to_know(r2,Conclusion)"/>
 </scope> </norm>
</normative-specification>
```

[3]That extension was inspired by [30].

The specification of constraints in the normative specification is commonly used by the literature, given that agents are supposed to be able to reason about the normative specification when they are playing some role in the multi-agent system. That is, with this approach the agents are almost directly able to reason about the argumentation schemes constraints as well. Another important point is that the multi-agent organisation, considering the extended normative specification, is able to monitor the use of arguments (instantiated by the argumentation schemes) among the agents. The main specification brings up some relations to the other (organisational) specifications in Soc$^{\mathtt{ARG}}$. Although the argumentation schemes, the structural and the functional specifications are independent and connected through the normative specification, as shown in Figure 3.1, we can define some relations between such specification, as detailed in the next sections.

3.1.1 Argumentation Schemes and the Structural Specification

The relation between the argumentation-scheme and the structural specifications is established, for the most part, in the normative specification, where the roles (from the structural specification) are linked to the argumentation schemes (in the argumentation-scheme specification) that are permitted/obligatory for each role. An indirect, but important, relation between the argumentation-scheme and the structural specifications is induced by the *communication link* between agents. Before any consideration of an agent having permissions/obligations to use certain argumentation schemes, clearly it will not use arguments instantiated from argumentation schemes (or any other kind of argument) towards agents with which it has no communication link. This consideration does not describe anything about the normative specification. Any agent can violate the norms in order to communicate with other agent using arguments. However, this communication is not possible without the communication link, because the infrastructure layer is supposed not to allow the exchange of messages by agents without a communication link.

Another relation between the argumentation-scheme and the structural specifications is the declaration of groups, where it is possible to specify the cardinality of each role within each group. The relation comes from the definition of groups where agents can make extensive use of argumentation in order to achieve the system's goals. This is a direct relationship with the permissions/obligation that each member of the group has to use argumentation schemes and the communication link between the members of the group.

Definition 7 (Argumentative Groups). *A group* gr *is an argumentative group iff there are agents* $ag_i, ag_j \in gr$ *playing some roles* $r_i, r_j \in \mathcal{SS}_R$ *and* $gr \in \mathcal{SS}_{rg}$*, where* $\langle n_i, dm, r_i, use(as_i, cons_i)\rangle, \langle n_j, dm, r_j, use(as_j, cons_j)\rangle \in \mathcal{NS}$*,* $dm \in \{\mathtt{permission}, \mathtt{obligation}\}$*, and* $link_{com}(r_i, r_j) \in \mathcal{SS}_{\sqsubset}$.

Other relations can arise from the argumentation-scheme and the structural specifications, which will depend on the domain. For example, the argumentation scheme described in Section 2.8 is directly related to the roles specified in the structural specification. Any argument instantiated from this scheme should be consistent with some concrete role from the structural specification.

3.1.2 Argumentation Schemes and the Functional Specification

Similarly to the structural specification, the relation between the argumentation-scheme and the functional specification is established, for the most part, in the normative specification. For example, the normative specification allows us to define the contexts where the argumentation schemes could be used in order to achieve particular goals or missions (from the functional specification). Thus, the functional specification is used to restrict the use of argumentation schemes, therefore any argumentation scheme that the agent role has permission/obligation to use could be used to achieve the goals the agent has committed to achieve, but argumentation schemes with context restrictions can be used only within a particular mission as specified.

Definition 8 (Contextual Argumentation Schemes). *An argumentation scheme is contextual for a mission m_i if $\langle as, \varphi, \mathcal{P}, \mathcal{CQ} \rangle \in \mathcal{AS}$ and $\langle n_i, dm, r_i, use(as, cons) \rangle \in \mathcal{NS}$ with $dm \in \{\texttt{permission}, \texttt{obligation}\}$, and $\langle m_i \rangle \in cons$ for some $m_i \in \mathcal{FS}_{\mathcal{M}}$.*

Contextual argumentation schemes are useful to restrict some types of argumentation schemes depending on the contexts determined by the functional specification. For example, formal dialogues could use stronger types of arguments as which we introduced in Section 2.8, while some other dialogues could use weaker argumentation schemes. Therefore, depending on the organisational goal to be achieved by the agents, they could be restricted by the normative specification to use only arguments instantiated from the schemes that are adequate for that context. Further, in some scenarios, the agents could be obliged to use some specific types of arguments.

3.1.3 Argumentation Schemes and the Normative Specification

As mentioned, the normative specification links the argumentation-scheme specification to the others (i.e., structural and functional specifications), including some indirect relations. However, besides the normative specification being used to link the schemes to the other specifications, it has itself some relations with the argumentation-scheme specification.

Definition 9 (Norm-Conforming Argumentation). *A Norm-Conforming Argumentation is an argumentation process (e.g., an ongoing instance of an argumentation-based dialogue protocol) that does not violate the normative specification, i.e., the participating agents only use arguments and critical questions instantiated from argumentation schemes that the agents are permitted to use, and satisfying also other constraints on such permissions, e.g. that the agents are playing the roles associated with the permission.*

The normative system used to regulate the behaviour of autonomous entities (agents) is not the aim of our work. Therefore, we will not discuss the internals of a normative system here. For our purposes, it suffices to understand that there is some normative system regulating the behaviour of the agents, enforcing the agents to follow the norms, for example by applying sanctions in order to try and stop agents from acting in (socially) undesirable ways [29].

We assume multi-agent systems where agents use standard languages to communicate and to represent arguments. Argumentation-based dialogues can be created based on a set of performatives (also called locutions or speech-acts) that the agents can use, and a protocol defining which moves/performatives are allowed at each step of the protocol based on some form of constraints [79, 80, 110]. One way to support agents in engaging in norm-conforming argumentation-based dialogues is defining a protocol that restrains the violation of norms, which are followed or not by agents in an autonomous way. In order to guide the definition of protocols that do not violate the normative specification of the Soc$^{\text{ARG}}$ model, we introduce some definitions below.

Definition 10 (Norm-Conforming Claims). *A claim ψ is norm-conforming iff $\langle as, \psi, \mathcal{P}, \mathcal{CQ}\rangle \in \mathcal{AS}$ and $\langle n_i, dm, \mathtt{role}(ag), use(as, cons)\rangle \in \mathcal{NS}$ with $dm \in \{\mathtt{permission}, \mathtt{obligation}\}$.*

Definition 11 (Norm-Conforming Questions). *A question φ is a norm-conforming question for ψ iff $\langle as, \psi, \mathcal{P}, \mathcal{CQ}\rangle \in \mathcal{AS}$, where $\langle cq_i, \varphi\rangle \in \mathcal{CQ}$, and $\langle n_j, dm, \mathtt{role}(ag), use(as, cons)\rangle \in \mathcal{NS}$ with $dm \in \{\mathtt{permission}, \mathtt{obligation}\}$ and $\langle cq_i, \varphi\rangle \notin cons$.*

3.1.4 Benefits of the Soc$^{\text{ARG}}$ Model

There are some clear benefits in using Soc$^{\text{ARG}}$ model, most of them resulting from the extended normative specification, and the shared argumentation schemes. In this section, we discuss the main benefits of the model.

Firstly, we argue that there are some benefits which come from the definitions presented in previous sections (namely Definitions 7, 8, and 9), whereby we are able to specify, in the Soc$^{\text{ARG}}$ specification: (i) groups of agents that are able to argue; (ii) contexts in which they could argue; and (iii) to guide agents to argue according to norm constraints. In (open) multi-agent systems, having this kind of “control” in the multi-agent specifications is rather valuable, given that different application domains could require different constraints in regards to communication. As an example, we could mention multi-agent applications recently developed on mobile systems. Normally, such an application domain has to restrict communication over the mobile network, using an architecture based on *personal* and *server* agents, where (i) *personal* agents are only responsible for collecting user information, sending it to corresponding *server*-side agents, and to interact with users by means of an interface, and (ii) *server* agents are responsible for most of the processing, decision-making, and (normally intensive) communication with other users' *server* agents. Examples of such systems are found in [47, 136, 27, 26]. Our approach allows for grouping agents and making them argumentative groups or not, depending on the application needs. Further, our approach allows specifying the contexts in which certain kinds of arguments could be permitted or prohibited, thus making the use of arguments specific for particular tasks in a multi-agent coordinated activity.

Secondly, besides the benefits related to the control of communication, our approach encourages argumentation schemes (reasoning patterns) to be shared knowledge within an agent system. Such an approach allows us to assume a more rational position for an agent in argumentation-based dialogues, where they are able to answer more critical questions by themselves. This occurs given the consideration of social/organisational components in the argumentation schemes, where such components are common knowledge to all agents. This is the main reason we have made this exercise of analysing the representation of argumentation scheme in the multi-agent system specification. Although we have proposed a model for specifying argumentation scheme in the multi-agent system specification, as mentioned before, our interest is argumentation scheme to be shared by agents in the system.

With argumentation schemes being shared by all agents in that society/organisation, and such argumentation schemes making references to components of the organisation, such as roles, authority link, etc., agents are (rationally) able to identify organisational information that is referred by argumentation schemes. For example, in the argumentation scheme *Argument from Position to Know*, the critical question **CQ4**:*"Is ag playing role R?"* is information that comes from the organisation. These references to organisational knowledge could bring some computational benefits to multi-agent system communication, as we will discuss later in this document. Of course, this benefit is inherent from the shared knowledge, and although our approach encourages such shared knowledge, it comes from organisation-based approaches that are typically used in multi-agent systems rather than from our model. However, in open multi-agent systems, even with agents knowing such information, they could still autonomously question that, overloading the system unnecessarily. In this respect, our model allows us to specify which agents are able to use such argumentation schemes (which is needed for them to argue), but constraining the use of critical questions which refer to organisational structure/components. For example as in $\langle n_1, permission, ag, use(as_1, cons)\rangle \in \mathcal{NS}$, with n_1 the norm identifier, as_1 the argumentation scheme identifier, the argumentation scheme *Argument from Position to Know*, and $cons$ an exception for not using the critical question **CQ4**.

The agents are able to violate the norms in order to benefit themselves or according to their intentions, and the normative system is supposed to apply sanctions when this occurs. This topic is not the subject of our work, and we argue that normative systems could be efficiently modelled in order to constraint undesired agent behaviour. In such cases, agents will have just norm-conforming argumentation-based dialogues, making/asking only norm-conforming claims and questions. Therefore, the argumentation-based communication will occur just as specified in the Soc$^{\text{ARG}}$ model, allowing a complete specification for multi-agents systems able to use argumentation techniques, which are already known for improving and enriching communication in such systems [127, 19, 126, 146, 68, 106]. Such specification allows more "control" over the communication that will occur in such systems. Further, it is possible to improve argumentation-based dialogues by means of high-level constraints for using the argumentation schemes, as well as by the shared knowledge, as argued above.

Although to have more control over communication is an interesting topic, it is not the main subject in our research, and it is part of our future planned work. The main point in our research is improving agent communication with argumentation-based techniques. The first improvement is very clear and already discussed in argumentation-based literature, i.e., *rich communication* [126]. The second improvement is related to agent rationality in argumentation-based communication using argumentation schemes. This means that, if agents are using reasoning patterns which are shared and known by all other agents in the multi-agent organisation, they can take some advantages from such a scenario. Agents could be able to communicate only the information that is required for a mutual understanding of such interactions, optimising agent communication.

3.2 Using Ontologies to Represent Argumentation Schemes

In [50, 49], we have proposed, in collaborative work, an infrastructure to support agents sharing knowledge towards semantics databases (web ontologies). In that approach, when developing a multi-agent system, we are able to specify the knowledge related to that application domain in OWL ontologies, and using the infrastructure proposed, agents are able to share that knowledge. Thus, agents use that infrastructure layer as a tool for storing, accessing and querying domain-specific OWL ontologies.

The main reason to represent shared knowledge in ontologies is to separate domain-dependent knowledge from a general architecture/structure for the development of a particular class (type) of applications, for example an architecture composed by *personal* and *server* agents [47, 25, 26, 27], which has been used to implement multi-agent systems to support task allocation between family members [106, 136]. That is, the same structure can be used to implement different applications, in which only the domain-dependent knowledge (the ontologies) might be replaced.

The infrastructure presented in [50, 49] is implemented in CArtAgO [132] and provides ontology features to agents by using the OWL API [61], which allows to create, manipulate and serialise OWL ontologies. Figure 3.2 shows an overview of the approach, where we have 3 workspaces with different configurations. Each workspace can have any number of instances of CArtAgO artifacts, and each artifact loads and encapsulates an OWL ontology. The agents can use their regular knowledge representation approach (e.g., belief base) simultaneous with the semantics databases, e.g., Workspace 1 in Figure 3.2, or completely replace the old approach, e.g., Workspace 2 in Figure 3.2. Workspace 3 of Figure 3.2 illustrates the usual approach, in which agents do not use the CArtAgO artifact to interact with ontologies.

Figure 3.2 clearly demonstrates that the approach presented in [50, 49] allows agents to share the same ontology, including agents from different workspaces (e.g., the agents on Workspace 1 and 2 are sharing the Ontology 2), as well as it allows the agents to use information from specific ontologies based on their role in the multi-agent system.

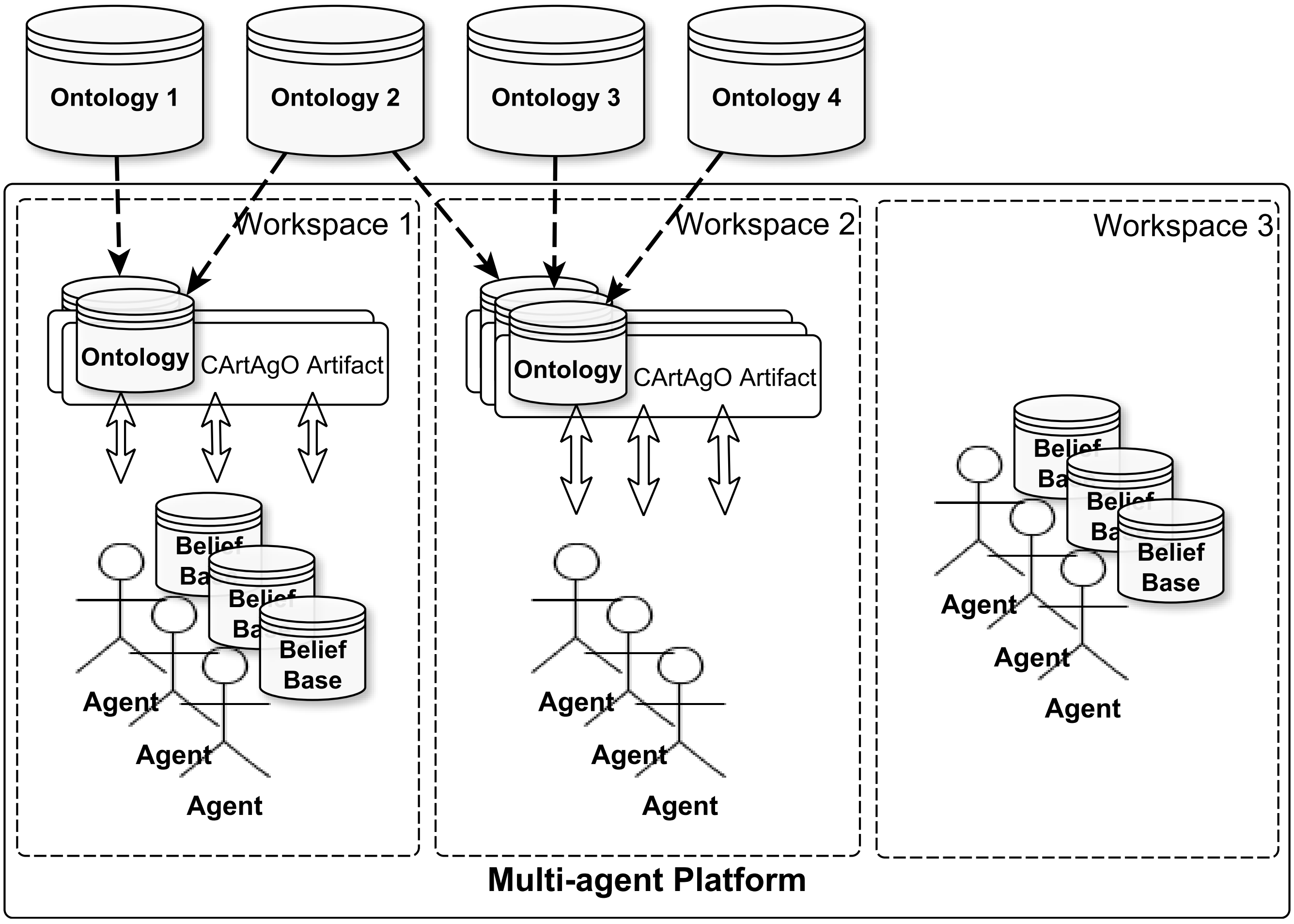


Figure 3.2 – Example of agents using shared ontologies (Workspace 1 and 2) versus usual multi-agent systems (Workspace 3) [50, 49].

3.3 Instantiating Argumentation Schemes

We have described two alternative infrastructure to represent argumentation schemes in multi-agent systems in Sections 3.1 and 3.2. For both approaches, either an extended organisational specification or shared ontologies, we are able to implement multi-agent systems in which agents share the argumentation schemes in a high level representation, as depicted in Figure 3.3. Thus, all agents in that multi-agent systems will be aware of those reasoning patterns for argumentation. In order to instantiate arguments from the argumentation schemes and, after that, to use those arguments in reasoning and communication, we propose that agents will have an internal representation of argumentation schemes, which can then be used by the inference mechanism we developed. Therefore, we proposed a particular representation of argumentation schemes using the language $\mathcal{L}$. In our approach, an argumentation schemes is represented as a defeasible inference rule $(p_0, \ldots, p_i \Rightarrow p_j)_{[sn]}$ plus the set of predicates representing the critical questions of the scheme $\{p_{k[sn]}, \ldots p_{l[sn]}\}$, in which sn is the name of the argumentation scheme being represented. When representing the argumentation schemes in the Jason Platform [33], the inference rule $(p_0, \ldots, p_i \Rightarrow p_j)_{[sn]}$ is represented using a special predicate $\texttt{defeasible\_rule}(\texttt{p}_\texttt{j}, [\texttt{p}_\texttt{0}, \ldots, \texttt{p}_\texttt{i}])_{[\texttt{sn}]}$,

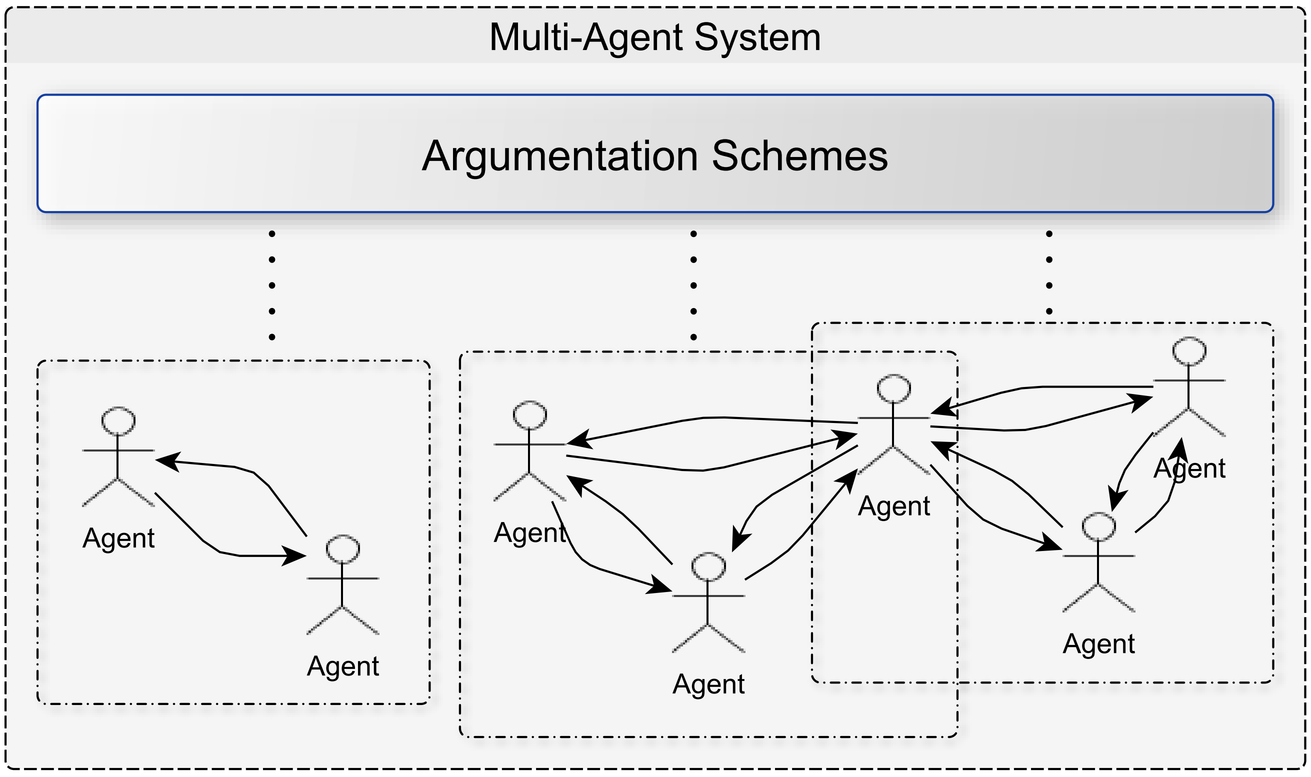


Figure 3.3 – Our approach to argumentation schemes in multi-agent systems.

and critical questions are represented using the special predicate cq(cq_n, p_k)$_{[sn]}$, with cq_n the critical question number. More details about the knowledge representation in Jason will be given in Chapter 4. For example, the argumentation scheme *role to know* is internally represented by agents as:

```
defeasible_rule(Conclusion,[asserts(Agent,Conclusion),
                role(Agent,Role),role_to_know(Role,Domain),
                about(Conclusion,Domain)])[as(as4rk)].

cq(cq1,role_to_know(Role,Domain))[as(as4rk)].
cq(cq2,honest(Agent))[as(as4rk)].
cq(cq3,asserts(Agent,Conclusion))[as(as4rk)].
cq(cq4,role(Agent,Role))[as(as4rk)].
```

In this representation, although we have four different predicates in the agent belief base, these predicates are linked towards the annotated argumentation schemes [as(as4rk)]. That means, for a particular instance of this reasoning pattern, for example, the variable `Role` in the critical questions `cq1` and `cq4` will have the same unification that the variable `Role` in the inference rule.

Using this internal representation for argumentation schemes, agents are able to instantiate arguments.

Definition 12 (Argument). *Formally, an argument is a tuple $\langle S, c\rangle^{\theta}_{\mathtt{sn}}$, where $\langle \mathtt{sn}, \mathcal{P}, \mathcal{C}, \mathcal{CQ}\rangle$ is the argumentation scheme used, θ is a most general unifier for the premises in $\mathcal{P}$ and the agent's current belief base, S is the set of premises and the inference rule of the scheme used to draw c (the conclusion of the argument). That is, S includes all instantiated premises from $\mathcal{P}$ — i.e., for all $p \in \mathcal{P}, p\theta \in \mathcal{S}$ — and the inference rule corresponding to the scheme ($\mathcal{P} \Rightarrow \mathcal{C}$); the conclusion c is the instantiation $\mathcal{C}\theta$ such that $S \models c$ (c can be inferred from S).*

For example, considering the argumentation scheme *role to know* introduced in Section 2.8, imagine a scenario in which all agents know that `john` (an agent in the system) is playing the role of `doctor` — `role(john, doctor)` — within the organisational structure. Further, the agents know that doctors know about cancer — `knows(doctor, cancer)`. Therefore, if `john` asserts that "*smoking causes cancer*" — `asserts(john, causes(smoking, cancer))`, and causes of cancer are a subject matter related to cancer — `about(causes(smoking, cancer), cancer)`, all agents are able to instantiate the argumentation scheme *role to know*, which allows them to conclude that smoking causes cancer: `causes(smoking, cancer)`. This argument is presented below:

```
⟨ { defeasible_rule(causes(smoking,cancer),
[asserts(john,causes(smoking,cancer)),role(john,doctor),
role_to_know(doctor,cancer), about(causes(smoking,cancer),cancer)]),
asserts(john,causes(smoking,cancer)),role(john,doctor),
role_to_know(doctor,cancer),about(causes(smoking,cancer),cancer)},
causes(smoking,cancer) ⟩[as(as4rk)]
```

Note that argumentation schemes are general structures for arguments, thus an agent is able to instantiate different arguments, for example $\langle S_1, c_1\rangle^{\theta_2}_{\mathtt{sn}}$ and $\langle S_2, c_2\rangle^{\theta_1}_{\mathtt{sn}}$, from the same argumentation scheme `sn`, when $\theta_1 \neq \theta_2$. Also, depending on the information available for an agent, it will be able to instantiate conflicting arguments from the same argumentation scheme. For example, imagine that another agent also playing the role of doctor, called `pietro`, asserts that "*smoking does not cause cancer*", i.e., `asserts(pietro, ¬causes(smoking, cancer))`. Any agent aware of both assertion, John's and Pietro's assertions, is able to construct conflicting arguments for `¬causes(smoking, cancer)` and `causes(smoking, cancer)`, instantiating the argumentation scheme *role to know*, both arguments attacking (in conflict with) each other (as we will see in the next chapter). However, the agents are able to question if `john` and `pietro` are honest (trustworthy, reliable) sources, if they really play the role of `doctor`, and the other questionable points indicated by critical questions in the argumentation scheme used[4]. Thus, when the critical questions related to that scheme are not positively answered, that instance of argument might be not acceptable. For example, if an agent has the information that "*Pietro is not a honest source*", i.e., `¬honest(pietro)`, that agent is not able to answer positively the critical question `honest(pietro)`, thus that instance

[4]Here, an important issue on the representation of argumentation schemes appears, in which not all critical questions refer to information explicitly represented on the structure of arguments instantiated from argumentation schemes.

of the argumentation scheme (i.e., that argument) might be not acceptable for that agent, i.e., the argument concluding $\neg\texttt{causes(smoking, cancer)}$ might be not an acceptable instance from the argumentation scheme *role to know*, while the argument concluding $\texttt{causes(smoking, cancer)}$ might be an acceptable instance from *role to know*, supposing that agent knows that $\texttt{john}$ is honest.

Thus, when considering argumentation schemes in multi-agent systems, the acceptance of a conclusion from an instance of an argumentation scheme, i.e., an argument instantiated from an argumentation scheme, is directly associated with the *critical questions* specified in that scheme. Critical questions may be asked before a conclusion from an argument to be accepted, and they point out to disputable information used in that argument [156]. Thus, when considering argumentation schemes, agents will not only need to instantiate the premises of the argumentation schemes available to them, i.e., agents need to match the information available to them with the premises of that argumentation scheme, but also they will need to consider if that argument is an acceptable instance from that argumentation scheme, i.e., they need to check if they are able to answer positively the critical questions associated with that argumentation scheme.

Definition 13 (Acceptable Instances of Arguments). *An argument $\langle S, c\rangle^{\theta}_{\mathrm{sn}}$, instantiated from an argumentation scheme $\langle \mathrm{sn}, \mathcal{P}, \mathcal{C}, \mathcal{CQ}\rangle$, is an acceptable instance of argument to an agent ag (where Δ_{ag} is its knowledge base) iff: (i) all premises in S can be drawn from ag knowledge base, i.e., $\forall p\theta \in \mathcal{S}, \Delta_{ag} \models p\theta$, either because p is asserted in its knowledge base, or because p is the conclusion of an acceptable argument; and (ii) all critical question related to the argumentation scheme $\langle \mathrm{sn}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$ are positively answered by ag, i.e., $\forall Cq_i \in \mathcal{CQ}, \Delta_{ag} \models Cq_i\theta$.*

Following our example, an agent also needs to have the information that $\texttt{john}$ is honest in order to have an acceptable instance of the argumentation scheme *role to know*, concluding that *"smoking causes cancer"*. For example, the belief base Δ represents the information[5] an agent needs to know in order to be able to instantiate an acceptable argument from the argumentation scheme *role to know*.

$$\Delta = \left\{ \begin{array}{l} \texttt{honest(john)} \\ \texttt{asserts(john, causes(smoking, cancer))} \\ \texttt{role(john, doctor)} \\ \texttt{role\_to\_know(doctor, cancer)} \\ \texttt{about(causes(smoking, cancer), cancer).} \end{array} \right\}$$

Besides instantiating arguments, which are acceptable instances of argumentation schemes, agents might have conflicting acceptable instances of arguments. Thus, agents need to check which arguments, given all arguments they are able to instantiate from the argumentation schemes available to them, are acceptable, given the attack relation between them.

[5]As we will describe in Section 4, $\texttt{honest(john)}$ could be implemented as an assumption, i.e., $\texttt{honest(Ag):-not(}\neg\texttt{honest(Ag))}$, describing that an agent $\texttt{Ag}$ could be considered honest if we are not able to show that it is not honest.

In order to allow an agent to execute this reasoning, we have proposed a general argumentation-based reasoning mechanism for Jason agents [101, 109]. After, we have extended this argumentation-based reasoning mechanism in order to allow agents to execute argumentation-based reasoning over acceptable instances of arguments from argumentation schemes, according to Definition 13. In the next chapter, we describe this part of our research.

4. ARGUMENTATION-BASED REASONING IN AGENT-ORIENTED PROGRAMMING LANGUAGES

In our research regarding argumentation-based reasoning, part of it published in [101, 109], we have investigated how to implement a general argumentation-based reasoning mechanism in Jason [33], given its particular language to represent agents' knowledge. In that research, we found strong evidence[1] that *defeasible logic* is a starting point to build argumentation systems. Such a claim can be confirmed by noting that most approaches for structured argumentation (cf. Section 2.7) use defeasible rules as part of their meta-language, as well as the claim made in [124]. Based on such evidence, we have implemented an argumentation-based reasoning mechanism in the Jason platform, using a particular formalisation for defeasible logic [93, 92], adapting its meta-interpreter implementation in Prolog called Defeasible-Prolog (d-Prolog for short) [91] for Jason [33]. The resulting argumentation-based reasoning mechanism has been presented in our recent publication [101], which we will describe in this chapter. We focus on both the formal specification of the argumentation framework as well as the knowledge representation and argumentation-based reasoning implementation on an agent-oriented programming language.

To the best of our knowledge, the only other strands of research that address argumentation-based reasoning in an agent-oriented programming language were introduced in [17, 150] and [68]. In [17], the author implements a separated module in Jason, thus agents need to query such a module and wait for the answer, making programming agents more difficult in practice. Our approach seems more adequate, given that the argumentation-based reasoning mechanism we developed is part of (i.e., integrated with) the agent reasoning cycle, and completely transparent for developers. In [150], the authors implement an argumentation-based reasoning mechanism in Jason platform using the Toulmin's model [149]. That approach is similar to [17], in the sense that agents in that approach need to execute a plan in order to calculate the status of a particular claim, given the set of warranties and refutation supporting and refuting that claim. After that, beliefs are added to the agents' belief base describing the status of that particular claim. That is, first agents execute a plan in order to know (belief) what is the status of a particular claim, given a predefined set of status, e.g., "certainly yes", then they are able to make decisions based on that new belief. In [68], the authors describe a preliminary version for argumentation-based negotiation (one-to-one negotiation) in JADE (Java Agent DEvelopment Framework) [13], where the agents negotiate resource allocation. Such work is based on the ABA Framework [46, 144]. The aim in [68] is to define communication policies which allow agents to provide reasons (arguments) for their refusal to provide the requested resources. Such communication policies can be implemented in our argumentation framework as well.

In order to present the argumentation-based reasoning mechanism, we divide this chapter into the following sections. First, we describe the way we represent agents' knowledge based on defeasible logic [93] and defeasible-Prolog [91] in AgentSpeak (in particular the AgentSpeak dialect

[1]In [109] we discuss such evidence.

available in the Jason platform [33]), including the adaptation of representation and additional possibilities for knowledge representation available in the Jason platform. Second, we describe the argumentation-based reasoning mechanism we developed in order to work with the new knowledge representation internally to the agent reasoning. After presenting the general argumentation-based reasoning mechanism, we introduce its extended version which allows agents to use argumentation schemes to instantiate arguments. Finally, we present an evaluation of the argumentation-based reasoning mechanism we developed.

4.1 Defeasible Knowledge Representation

Defeasible logic [93] and defeasible-Prolog [91] have a particular representation in the form of facts, rules, and a "superiority" relation. In order to adequately represent this in Jason we represent facts, inference rules, and the superiority relation as belief predicates that are treated similarly to other information in the agent's belief base. So the knowledge representation in our framework is organised as follow:

- **facts**: are represented as predicates, so for example "Bob is a graduate student" is represented as a simple predicate such as `grad_student(bob)`;
- **strict rules**: are represented as a special predicate `strict_rule(Head,Body)`, so for example "graduate students are students" is represented as `strict_rule(student(X),grad_student(X))`; the body can also be a list of predicates to represent conjunction;
- **defeasible rules**: are represented as a special predicate `defeasible_rule(Head,Body)`, so for example "graduate students usually study hard" is represented as `defeasible_rule(studies_hard(X),grad_student(X))`; as above, the body can also be a list rather than a single predicate; and
- **superiority relation**: the superiority relation is represented as `sup(`$\mathtt{Rule}_1$`,`$\mathtt{Rule}_2$`)` stating that $\mathtt{Rule}_1$ is superior to $\mathtt{Rule}_2$.

Further, in order to maintain the coherence when we construct inference rules using first-order predicates, as is our case, we allow formulæ such as `X\==Y` into body of rules (i.e., the term instantiated to `X` is different from the one with which `Y` is instantiated), `X==Y` (i.e., equality), and `X>Y` etc[2]. into the list, allowing us to constrain the instantiated variables (within the same predicate or not).

[2]Such formulae also allows us to implement value-based argumentation [15] .

In argumentation-based reasoning, as well as in any argumentation system, an important issue related to knowledge representation is the representation of conflicting information. Regarding conflicting information, we can consider different types of opposition, the most common being *negation* where the negation of a proposition is its opposite in a strong sense of opposition — they are in direct contradiction, for example, "This pen is black" and "This pen is not black". Besides, there is also a weaker sense of opposition, called contraries, for example, "This pen is black" and "This pen is green" [156]. In our argumentation framework, we consider both types of opposition. The strong type, *negation*, is represented using the usual symbol[3] "$\neg$", so "Bob is not a graduate student" is represented as $\neg$`grad_student(bob)`. While the weak type of opposition is represented using a special predicate `comp(good,bad)`, meaning that "good" is contrary to "bad". Therefore, considering the two types of opposition, the notion of conflicting information in our framework can be defined.

Definition 14 (Conflicting Information). *Two pieces of information p_i and p_j from $\mathcal{L}$ are said in conflict when:*

- *p_i and p_j are the negation of each other, e.g., p_i =* `causes(smoking, cancer)` *and p_j =* $\neg$`causes(smoking, cancer)`*, or;*
- *p_i and p_j are semantically declared as conflicting information, i.e., p_i =* `reliable(john)` *and p_j =* `unreliable(john)`*, with some declaration of complement, conflict, or contrary information, e.g.,* `comp(reliable(john),unreliable(john))`*.*

For both cases we adopt a general operator for conflict $\overline{p_i}$, in which $p_j \equiv \overline{p_i}$ means that p_i and p_j are conflicting information.

Having described how we represent individual pieces of information in our framework, we are now able to describe how arguments are created on top of those. Although the construction of arguments is related to the argumentation-based reasoning mechanism that we will describe in the next section, we describe here the representation of arguments themselves. It is important to note that our argumentation-based reasoning mechanism constructs an argument looking for the inference rules and facts available in the agent's belief base at that particular time in order to derive a specific conclusion from them. Therefore, arguments are composed of a set of beliefs — representing the facts and inference rules that can be used as support — and that particular conclusion supported/derived from that set of beliefs.

Definition 15 (Argument). *An argument is a tuple $\langle S, c\rangle$, where S is a set of beliefs representing facts and inference rules which supports a conclusion c.*

For example, the classical defeasible argument called *argument from perception*, introduced by Pollock [122], says that:

[3]In Jason, the corresponding symbol for strong negation is "$\sim$".

"When an object looks red, then (normally, but subject to exceptions) it is red, and this object looks red to me, therefore this object is red."

This argument, concluding that some particular object is red, is represented as $\langle S, \texttt{color(obj,red)}\rangle$ where S is the set of beliefs representing facts/predicates and inferences rules used to draw that conclusion, as follows:

[`defeasible_rule(colour(obj,red), looks(obj,red)),looks(obj,red)`]

meaning that the agent believes, through its sensing capabilities, that the object looks red — it has a belief `looks(obj,red)` in its belief base — and the agent has a defeasible rule `defeasible_rule(colour(X,Y),looks(X,Y))` whereby it can infer that any object/thing that looks like being of some particular colour, can be, at least tentatively, inferred to have that particular colour, i.e., in that instance the agent can presumably conclude that object to be red (given also that there is no contrary information to conclude so). Later, in this section, we will discuss this kind of *reasoning patterns* in multi-agent systems in more detail.

Similarly to ABA, our approach has its root in logic programming, thus it has the advantage of a fine level of granularity, given it offers an interleaving of construction of argument and determining their acceptability [46], as we will see in the next section.

4.2 Argumentation-Based Reasoning Mechanism

The argumentation-based reasoning mechanism, as mentioned before, is used by agents to query the acceptability of conclusions (and their supporting arguments) in their belief bases; such queries are executed internally during the reasoning process. When a conclusion is queried internally by an agent, as is usually the case, its reasoning mechanism searches for that information in their belief base, including acceptable inferences through the extensions we developed.

Before we introduce acceptable arguments (i.e., a conclusion supported by acceptable inferences given the state of an agent's belief base) we need to discuss a few issues. Intuitively, in our argumentation framework we have two kinds of arguments, where arguments that use only *strict* rules are stronger than arguments that use any *defeasible* rules: (i) **strict arguments** are formed only by facts and strict rules (i.e., indisputable knowledge). It is assumed that the strict part of any knowledge base (in practice the agent's belief base) is consistent (i.e., contradictions cannot be derived); (ii) **defeasible arguments** are created using at least one defeasible rule (corresponding to the points of weakness of the argument).

An important aspect of this argumentation framework is that the strict part of the knowledge in the belief base of agents is assumed to be consistent. This is because the strict knowledge is composed by beliefs that can be determined as factual and strict inference rules which are all indisputable knowledge (i.e., they correspond to an agent's knowledge rather than its beliefs), therefore strict knowledge can be naturally assumed to be consistent.

Definition 16 (Consistency of Strict Knowledge). *Let Δ^{strict} be the strict part of an agent's belief base Δ. A belief base is said to be consistent if there is no contradictory information derivable from Δ^{strict}, i.e., $\Delta^{strict} \models \varphi$ and $\Delta^{strict} \models \overline{\varphi}$ do not hold simultaneously at any given time.*

Whenever an agent queries the acceptability of a particular conclusion, the internal argumentation-based reasoning mechanism developed (extending the normal agent reasoning mechanism) tries to find a rule for that particular queried conclusion and then it attempts to prove the premises of that rule as usual in backwards chaining. In summary, the reasoning mechanism tries recursively to find a proof for that particular conclusion from the beliefs in the belief base, including rules, facts, and assumptions.

In this process, the reasoning mechanism considers conflicting arguments (inferences). Considering a dialectical point of view, two types of attack (conflict) between arguments can be considered, according to [156]: (i) a strong kind of conflict, where one party has a thesis to be proved, and the other part has a thesis that is the opposite of the first thesis, and (ii) a weaker kind of conflict, where one party has a thesis to be proved, and the other part doubts that thesis, but has no opposite thesis of his own. In the strong kind of conflict, each party must try to refute the thesis of the other in order to win. In the weaker form, one side can refute the other, showing that their thesis is doubtful. This difference between conflicts are inherent from the structure of arguments, and can be found in the work of others, e.g. [123].

Both types of conflicts, the strong and the weaker kinds of conflict, are also considered in monological argumentation frameworks. On the one hand, the strong kind of conflict refers to two arguments supporting conflicting conclusions, in which each argument has its own set of evidence, i.e., its support. On the other hand, the weaker kind of conflict refers to an argument attacking (in conflict with) part of the support of another argument, i.e., that argument does not support that the conclusion is false, but it supports that some information used as the support of that conclusion is not true.

Thus, in order to define the acceptability of an argument (and the respective queried conclusion) we need to consider conflicting arguments. Conflict between arguments are of two kinds:

Definition 17 (Attack Between Arguments). *Let $\langle S_1, c_1 \rangle$ and $\langle S_2, c_2 \rangle$ be two arguments. Attacks between arguments can be generalised as:*

- *The argument $\langle S_1, c_1 \rangle$ rebuts the argument $\langle S_2, c_2 \rangle$ iff $c_1 \equiv \overline{c_2}$.*
- *The argument $\langle S_1, c_1 \rangle$ undercuts the argument $\langle S_2, c_2 \rangle$ iff $c_1 \equiv \overline{c_3}$ for some $\langle S_3, c_3 \rangle$, where $S_3 \subseteq S_2$.*

When two arguments are in conflict, i.e., the arguments attack each other, this does not necessarily mean that one argument defeats the other. Defeat is a "successful" attack, and it considers the set of arguments that defend each other, including preferences between the conflicting arguments [156]. In our framework, the set of acceptable arguments from an agent's belief base is defined in terms of the ***defeasible semantics*** introduced in [57]. The defeasible semantics is similar to the ***grounded semantics*** from Dung's work [45], and it is based on the so-called ***pre-empting defeaters*** [91]. The preempting defeaters of [91] are called ***ambiguity blocking*** (in regards to the argumentation system) in [57]. This means that defeasible arguments that are rebutted by as strong as, or stronger, arguments (defined through some sort of preference that will be explained below) are no longer available to rebut other arguments. An example of preempting defeaters is the knowledge base represented by Δ below, where we use $\Rightarrow$ to refer to ***defeasible*** inferences:

$$\Delta = \left\{ \begin{array}{lll} a & a \Rightarrow b & b \Rightarrow c \\ x & x \Rightarrow e & e \Rightarrow \neg c \\ y & y \Rightarrow \neg e & c \Rightarrow d \end{array} \right\}$$

In this example, considering Δ as the belief base of some agent, the agent may conclude d using $\{a,\ a \Rightarrow b,\ b \Rightarrow c,\ c \Rightarrow d\}$ as support. Although there is an argument to $\neg c$ supported by $\{x,\ x \Rightarrow e,\ e \Rightarrow \neg c\}$ which rebuts the sub-argument for d that concludes c (undercutting the first argument), this argument (the argument that derives $\neg c$) is defeated (by undercut) by an argument with support $\{y,\ y \Rightarrow \neg e\}$ which prevents the use of that argument to attack the argument for d.

Although in the example above we have an acceptable argument for d, the arguments with support $\{y,\ y \Rightarrow \neg e\}$ and $\{x,\ x \Rightarrow e\}$ (which is a sub-argument for c in the example) are in conflict, and the argumentation-based reasoning mechanism[4] is not able to decide which one is acceptable, i.e., both are treated as unacceptable. A way to deal with undecided conflicts is to use preferences over the arguments.

Clearly, strict arguments are stronger than defeasible arguments and they have priority, i.e., when arguments are in conflict, strict arguments always defeat defeasible ones. Considering only defeasible arguments, in our framework we have two types of priority[5]: (i) priority by specificity, which is originally defined in defeasible logic [93], and (ii) the explicit declaration of priority between defeasible rules, using a special predicate in the knowledge representation of our argumentation framework. In priority by specificity, more specific conclusions have priority over more general ones. To exemplify this idea, consider the well-known Tweety example implemented in a Jason agent:

```
defeasible_rule(flies(X),bird(X)).
defeasible_rule(¬flies(X),penguin(X)).
defeasible_rule(bird(X),penguin(X)).
penguin(tweety).
```

All clauses in the example are defeasible rules. Considering the knowledge above, we have two conflicting arguments, one supporting that Tweety flies: "Tweety flies, because it is

[4]This characteristic is from the original implementation of defeasible Prolog [91], and it is what gave rise for the name *ambiguity blocking* in [57].

[5]Although we present here the original argumentation framework we developed, we are working on an extension which considers more kinds of priority. The preliminary work is found in [82, 83, 107].

a penguin, penguins are birds, and birds fly", and one supporting that Tweety does not fly: "Tweety does not fly, because it is a penguin and penguins do not fly". Our argumentation-based reasoning mechanism (as well as defeasible-Prolog [91] on which our reasoning mechanism is based) concludes, in this case, that Tweety does not fly, because the rule for penguins `defeasible_rule(¬flies(X),penguin(X))` is more specific than a rule for birds `defeasible_rule(flies(X),bird(X))`, given that penguin is a subclass of birds, represented by `defeasible_rule(bird(X),penguin(X))`. In this manner, the argument for Tweety does not fly, `¬flies(tweety)`, has priority over the other one and so defeats it[6].

Furthermore, when there exist two rules deriving contradictory information, the language used in our approach, as described before, allows us to declare, in an explicit way, priority between these rules, using a special predicate `sup(Rule1,Rule2)`, indicating that inferences using `Rule1` have priority over inferences using `Rule2`. Therefore, when two conflicting arguments are constructed using these conflicting rules, this declaration is used in order to decide which conclusion will actually be derived. Therefore, we can define the acceptability of an argument as follows (based on [57]):

Definition 18 (Acceptable Arguments). *An argument $\langle S, c\rangle$ is acceptable to an agent ag (where Δ_{ag} is its belief base) if $\langle S, c\rangle$ is finite, and: (a) $\langle S, c\rangle$ is strictly inferred, or (b) every argument attacking $\langle S, c\rangle$ is defeated by some argument $\langle S_n, c_n\rangle \in \Delta_{ag}$ (i.e., all arguments that attack $\langle S, c\rangle$ cannot be inferred from Δ_{ag} because they are attacked by as strong as, or stronger, arguments in Δ_{ag} so they are not acceptable in Δ_{ag}).*

The reasoning mechanism extending the usual agent reasoning mechanism in Jason in accordance with the argumentation-based reasoning mechanism that we proposed was implemented by adapting d-Prolog [91], using the Prolog-like rules which are interpreted by Jason with some limitations (e.g., the "cut" operator is not available). The implementation is based on logic programming and the formal semantic and syntax of the AgentSpeak language extension that can be found in [33]. A part of that reasoning mechanism is presented below:

```
strict_der(Content):- Content.
strict_der([Content]):- strict_der(Content).
strict_der([First|Rest]):- strict_der(First) & strict_der(Rest).
strict_der(Content):- strict_rule(Content,Cond) & strict_der(Cond).
```

In this part of the implementation, we demonstrate the derivation of *strict inferences*, where first we check if the queried content is a belief or a formula (i.e., `X\==Y`, `X==Y`, etc.), then if it is a list of a single element, then if it is a list of more than one element, and finally if it is the `Head` of a strict rule and if the `Condition` (which derives the `Content`) is also strictly

[6]We have extended, in a parallel work, the preferences in the argumentation-based reasoning mechanism presented here, considering meta-information available in the Jason Platform, e.g., the source and the time that the information has been acquired by the agent; such extensions are presented in [82, 83, 107].

derived. These rules permit the agent to query if a content is strictly derived in its knowledge base (remember that strict knowledge is indisputably known).

In our framework, when an agent wants to query if it has an argument to support some conclusion, it uses the special predicates `strict_der(Content)` and `def_der(Content)` depending on whether the agent needs that information to be strictly or defeasibly inferred, respectively. When an agent needs the argument (set of beliefs representing facts and inference rules used in that acceptable inference) to support a claim in a dialogue or a decision-making process, the argument is accessible using a second parameter in the query, which we call `Arg`. Each rule and fact used in the inference of that particular query are stored using the internal action `.concat` (which concatenates a list with the new element, e.g., a rule or a fact) available in Jason. Thus, depending on the strategy of the agent, it can verify if it has a strict or defeasible argument, using `strict_der(Arg,Content)` and `def_der(Arg,Content)`, or if this distinction is not necessary, the agent can use the special predicate `argument(Content,Arg)` inferred by:

```
argument(Content,Arg):- strict_der(Arg,Content)|def_der(Arg,Content).
```

where we check first if the content is inferred in a strict way, and then if the content can be inferred in a defeasible way.

The argumentation-based reasoning mechanism queries the acceptability of arguments at runtime, and given the way the agent reasoning cycle works, the arguments are constructed using the most up-to-date information available to the agent given a snapshot of its belief base. Therefore, the acceptability of that particular argument (supporting the queried conclusion) is guaranteed to be in accordance with the updated information available for that agent at the moment of the query. Of course, because of the dynamism of typical multi-agent systems, at the very next reasoning cycle that same argument may no longer be derivable for that same agent.

4.3 Example

As an example, we use the ***paper submission scenario*** from [109, 101]. In our example, imagine that an agent has submitted a "paper" to BRACIS 2016. The agent believes that its paper is good and will be accepted, so it commits itself to buy a ticket to Recife because it concludes `go_to(recife)` from its belief base, given that BRACIS 2016 is to take place in Recife, i.e., it has the belief `held_in(bracis,recife)`.

```
defeasible_rule(go_to(L),[held_in(C,L),accepted(P,C)]).
defeasible_rule(accepted(P,C),[submitted(P,C),good(P)]).
submitted(paper,bracis).
good(paper).
held_in(bracis,recife).
```

A plan that the agent could use to buy a ticket for some location instantiated by `L` has the following format (in the Jason platform):

```
+!buyTicket(L) : def_der(go_to(L))
          <-  buyTicket(L).
```

meaning that if the agent has reasons to believe that it needs to go to some location `L`, it may buy a ticket to that location. When the agent needs the actual argument that supports a particular decision, it could use `argument(go_to(L),Arg)`, thereby instantiating `Arg` with such argument, which could then be used to justify such decision-making to other agents, for example.

However, before the agent buys its ticket, it checks the BRACIS 2016 webpage and realises that the page limit for BRACIS papers is 6 pages (including references) and the agent has, unfortunately, submitted a longer paper than allowed. In addition to the allowed paper length, the agent has the information that papers longer than allowed are *strictly* not accepted. This knowledge is represented as follows:

```
strict_rule(¬accepted(P,C),[longer_for(P,C),submitted(P,C)]).
strict_rule(longer_for(P,C),[paper_length(P,X),allowed_length(C,Y),X>Y]).
paper_length(paper,9).
allowed_length(bracis,6).
```

With the new information the agent can no longer conclude `go_to(recife)`, considering that the new information allows the inference of a strict argument for `¬accepted(paper,bracis)`, which defeats the argument for `go_to(recife)`, i.e., the argument for `¬accepted(paper,bracis)` successfully undercuts the argument for `go_to(recife)`, considering that they are in conflict and strict arguments have priority over defeasible ones. Therefore, the plan above is no longer *applicable* (the plan's context is no longer satisfied).

4.4 Reasoning with Argumentation Schemes

In Chapter 3, we described how we represent argumentation schemes in a high-level infrastructure into multi-agent systems in order to share those reasoning patterns to all agents in the system. Also, we described how agents internally represent those reasoning patterns in order to instantiate arguments they might use during reasoning and dialogues.

As described, for example, the argumentation scheme *role to know* is internally represented by agents as:

```
defeasible_rule(Conclusion,[asserts(Agent,Conclusion),
                role(Agent,Role),role_to_know(Role,Domain),
                    about(Conclusion,Domain)])[as(as4rk)].

cq(cq1,role_to_know(Role,Domain))[as(as4rk)].
cq(cq2,honest(Agent))[as(as4rk)].
cq(cq3,asserts(Agent,Conclusion))[as(as4rk)].
cq(cq4,role(Agent,Role))[as(as4rk)].
```

Further, these predicates are linked towards the annotated argumentation schemes [as(as4rk)]. Thus, when agents instantiate this reasoning pattern, the variables used in the predicates will have a corresponding unification.

In order to allow agents instantiating and reasoning using arguments from argumentation schemes, we have extended the argumentation-based reasoning mechanism introduced in this chapter. Thus, agents first instantiate argumentation schemes and check if they are acceptable instances of arguments, according to Definition 13, then they are able to use those acceptable instances of arguments in order to check which ones are acceptable, considering all arguments available to those agents, i.e., checking attacks between arguments.

For example, imagine that an agent ag has the following knowledge base:

$$\Delta_{ag} = \left\{ \begin{array}{l} \texttt{honest(john)} \\ \texttt{honest(pietro)} \\ \texttt{asserts(john, causes(smoking, cancer))} \\ \texttt{asserts(pietro,} \neg \texttt{causes(smoking, cancer))} \\ \texttt{role(john, doctor)} \\ \texttt{role(pietro, doctor)} \\ \texttt{role\_to\_know(doctor, cancer)} \\ \texttt{about(causes(smoking, cancer), cancer).} \end{array} \right\}$$

Thus, ag is able to instantiate 2 different acceptable instances of arguments from the argumentation scheme *role to know*, according Definition 13:

```
⟨{role(john,doctor), role_to_know(doctor,cancer),
asserts(john,causes(smoking,cancer)),
about(causes(smoking,cancer),cancer),
(role(Agent,Role), role_to_know(Role,Domain),
    asserts(Agent,Conclusion),
    about(Conclusion,Domain) ⇒ Conclusion)[as(role_to_know)],
causes(smoking,cancer)⟩.

⟨{role(pietro,doctor), role_to_know(doctor,cancer),
asserts(pietro,¬causes(smoking,cancer)),
about(¬causes(smoking,cancer),cancer),
(role(Agent,Role), role_to_know(Role,Domain),
    asserts(Agent,Conclusion),
    about(Conclusion,Domain) ⇒ Conclusion)[as(role_to_know)],
¬causes(smoking,cancer)⟩.
```

Considering that all critical questions are positively answered by the agent *ag*, both arguments are acceptable instances of arguments, but they are in conflict with each other in their conclusions. Thus, considering that an agent has only those arguments, i.e., one argument for and another argument against the conclusion that causes(smoking, cancer), it is not able to decide that conflict and both arguments are not acceptable to it. This is a rational position, given the agent has evidence supporting both conflicting conclusions.

However, when more information became available to the agent *ag*, the status (acceptability) of arguments might change. For example, imagine that pietro is not actually a doctor, i.e., role(pietro, doctor) is not true, and the agent *ag* becomes aware of that information towards the hospital director named mathew. Considering that the agent *ag* also has the following information about mathew in its knowledge base:

$$\Delta_{ag} = \left\{ \begin{array}{l} \texttt{honest(mathew)} \\ \texttt{asserts(mathew, role(pietro, nurse))} \\ \texttt{role(mathew, hospital\_director)} \\ \texttt{role\_to\_know(hospital\_director, employee)} \\ \texttt{about(role(pietro, nurse), employee).} \end{array} \right\}$$

Thus, *ag* is able to construct another acceptable instance of argument from the argumentation scheme ***role to know*** as follows:

```
⟨{role(mathew,hospital_director),
role_to_know(hospital_director,employee),
asserts(mathew,role(pietro,nurse)),
about(role(pietro,nurse),employee),
(role(Agent,Role), role_to_know(Role,Domain),
    asserts(Agent,Conclusion),
    about(Conclusion,Domain) ⇒ Conclusion)[as(role_to_know)],
role(pietro,nurse)⟩.
```

This argument attacks the argument concluding that ¬causes(smoking, cancer), at role(pietro, doctor), considering that role(pietro, nurse) is in conflict with role(pietro, doctor). Thus, this argument is defeated and cannot be used to defeat the argument concluding that causes(smoking, cancer). That means, we have extended the definition of acceptable arguments as follows:

Definition 19 (Acceptable Arguments). *An argument $\langle S_1, c_1 \rangle^{\theta}_{\texttt{sn}}$ is acceptable for an agent if it is an acceptable instance of argument from an argumentation scheme* sn, *and it is not defeated by any other acceptable argument. That means, considering the set of arguments an agent is able to construct from its knowledge base, an acceptable argument either is not attacked by any other argument, or when it is attacked by another argument, this another argument is attacked by an acceptable argument.*

Considering that the three arguments above are acceptable instances of arguments from the argumentation scheme *role to know*, the argument concluding that $\texttt{causes(smoking, cancer)}$ is attacked by the argument concluding $\neg\texttt{causes(smoking, cancer)}$, which is attacked by the argument concluding $\texttt{role(pietro, nurse)}$. Considering that the argument concluding $\texttt{role(pietro, nurse)}$ is not attacked by any other argument, $\texttt{role(pietro, nurse)}$ and $\texttt{causes(smoking, cancer)}$ are the two acceptable conclusions (and their respective arguments) in our example.

4.5 Evaluation

The main characteristic we desire in our framework is *generality*. In order to evaluate its generality, we have selected some argumentation schemes from different areas of study in the literature. We have represented those reasoning patterns in our framework, and then we evaluated different scenarios of argumentation-based reasoning using them. In Section 4.5.1, we discuss the argumentation schemes selected to this evaluation, how they are represented in our framework, and some examples of reasoning that agents can execute using them.

Another interesting exercise we have made to evaluate our framework for argumentation schemes in multi-agent systems was the modelling of argumentation schemes for a particular class of multi-agent systems applications. That is, we exercise the process of evaluating a particular application domain (problem) and then to model (or identify) argumentation schemes required to that application domain. In this exercise, we have analysed the problem of data access control between multi-agent systems applications, then we proposed argumentation schemes that argumentation-based interface agents could use to make decisions on when to share or not information from its multi-agent system. In section 4.5.2, we present this research.

Finally, although it is not one of our main goal in this research, we have evaluated our implementation in order to check how fast agents are able to reach (or not) a particular conclusion using our extended argumentation-based reasoning mechanism, and how much the critical questions influence the time necessary to agents reach a conclusion. In Section 4.5.3, we present these results.

4.5.1 Representation and Reasoning with Argumentation Schemes from the Literature

Argumentation Scheme from Role to Know: The first argumentation scheme we have represented in our framework was the argumentation scheme from role to know, which is an adapted version from the argumentation scheme for position to know [153, 156]. Also, the argumentation scheme for position to know is the argumentation scheme most used by the literature when exemplifying what argumentation schemes are. Here, making reference to multi-agent system concepts, i.e., roles. This argumentation scheme has been extensively used to exemplify our framework in the previous sections, and it has been successfully represented in our framework. It allows agents to construct different arguments based on assertions coming from experts on a particular subject. After that, agents are able to use those arguments to reach well supported conclusions (when they exist), as we showed in Section 4.4.

Argumentation Schemes from Biomedical Domain: The second scenario used to evaluate the generality of our framework comes from a biomedical paper [58], which have extracted seven causal argumentation schemes from biomedical research domain in order to use them in argument mining. Although the main goal in [58] is to model argumentation schemes that could be used to automatise argument mining from research papers in the biomedical domain, we have modelled such argumentation scheme in order to allow agents to reason about such a domain.

The corresponding argumentation schemes and their representation in our framework are described below:

- **Method of Agreement**: This argumentation scheme says that *if a group* `G` *has a phenotype* `P` *and a genotype* `M`*, then that genotype* `M` *causes the phenotype* `P`.

  This argumentation scheme has been represented in our approach as: sn = `ma_paper01`, $\mathcal{P} = \{\texttt{have\_phenotype(G,P)}, \texttt{have\_genotype(G,M)}\}$, $\mathcal{C} = \texttt{cause(M,P)}$, $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

  ```
  defeasible_rule(cause(M,P),[have_phenotype(G,P),
      have_genotype(G,M)])[as(ma_paper01)].
  ```

- **Method of Failed Agreement (no effect)**: This argumentation scheme says that *if a group* `G` *does not have phenotype* `P` *and has the genotype* `M`*, then the genotype* `M` *does not cause the phenotype* `P`.

  This argumentation scheme has been represented in our approach as: sn = `mfa_paper01`, $\mathcal{P} = \{\neg\texttt{have\_phenotype(G,P)}, \texttt{have\_genotype(G,M)}\}$, $\mathcal{C} = \neg\texttt{cause(M,P)}$, $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

  ```
  defeasible_rule(¬cause(M,P),[¬have_phenotype(G,P),
      have_genotype(G,M)])[as(mfa_paper01)].
  ```

- **Method of Difference**: This argumentation scheme says that *if a group* `G1` *has a phenotype* `P` *and has the genotype* `M`*, a group* `G2` *does not have the phenotype* `P` *and does not have the genotype* `M` *then the genotype* `M` *causes the phenotype* `P`*.*

  This argumentation scheme has been represented in our approach as: sn = md_paper01, $\mathcal{P}$ = $\{$have_phenotype(G1, P), have_genotype(G1, M), $\neg$have_phenotype(G2, P), $\neg$have_genotype(G2, M), (G1\ == G2)])$\}$, $\mathcal{C}$ = cause(M, P), $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

  ```
  defeasible_rule(cause(M,P),[have_phenotype(G1,P),
       have_genotype(G1,M),¬have_phenotype(G2,P),
       ¬have_genotype(G2,M),(G1\ ==G2)])[as(md_paper01)].
  ```

- **Analogy**: This argumentation scheme says that *if a group* `G1` *has a phenotype* `P1`*, a group* `G2` *has a phenotype* `P2`*, where* `P2` *is similar to* `P1`*, the group* `G1` *has the genotype* `M1`*, the group* `G2` *has the genotype* `M2`*, where* `M2` *is similar to* `M1`*, and the genotype* `M1` *causes the phenotype* `P1` *then the genotype* `M2` *causes the phenotype* `P2`*.*

  This argumentation scheme has been represented in our approach as: sn = a_paper01, $\mathcal{P}$ = $\{$have_phenotype(G1, P1), have_phenotype(G2, P2), similar(P1, P2), have_genotype(G1, M1), have_genotype(G2, M2), similar(M1, M2), cause(M1, P1), (G1\ == G2)])$\}$, $\mathcal{C}$ = cause(M2, P2), $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

  ```
  defeasible_rule(cause(M2,P2),[have_phenotype(G1,P1),
       have_phenotype(G2,P2),similar(P1,P2),have_genotype(G1,M1),
       have_genotype(G2,M2),similar(M1,M2),cause(M1,P1),
       (G1\ ==G2)])[as(a_paper01)].
  ```

- **Eliminate Difference**: This argumentation scheme says that *if a group* `G1` *has a phenotype* `P`*, a group* `G2` *does not have the phenotype* `P`*, the group* `G1` *has the genotype* `AB`*, the group* `G2` *has the genotype* `B`*, the genotype* `A` *is the difference between* `AB` *and* `B`*, then the genotype* `A` *causes the phenotype* `P`*.*

  This argumentation scheme has been represented in our approach as: sn = ed_paper01, $\mathcal{P}$ = $\{$have_phenotype(G1, P), have_genotype(G1, AB), $\neg$have_phenotype(G2, P), difference(AB, B, A), have_genotype(G2, B), (G1\ == G2)])$\}$, $\mathcal{C}$ = cause(A, P), $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

  ```
  defeasible_rule(cause(A,P),[have_phenotype(G1,P),
       have_genotype(G1,AB),¬have_phenotype(G2,P),
       difference(AB,B,A),have_genotype(G2,B),
       (G1\ ==G2)])[as(ed_paper01)].
  ```

- **Consistent Explanation**: This argumentation scheme says that *if a group `G1` has a genotype `M1`, it has a protein `Prot`, and it has a phenotype `P1`, `M1` causes `Prot`, `Prot` is associated with the phenotype `P1`, `M1` causes `P1`, a group `G2` has the genotype `M2`, where `M2` is similar to `M1`, `G2` has the protein `Prot`, `G2` has the phenotype `P2`, where `P2` is similar to `P1`, `M2` causes `Prot`, `Prot` is associated with `P2`, then the genotype `M2` causes the phenotype `P2`.*

  This argumentation scheme has been represented in our approach as: sn = ce_paper01, $\mathcal{P}$ = {similar(M1,M2), similar(P1,P2), , have_genotype(G1,M1), have_protein(G1,Prot), have_phenotype(G1,P1), cause(M1,Prot), assoc(Prot,P1), have_genotype(G2,M2), have_protein(G2,Prot), have_phenotype(G2,P2), cause(M2,Prot), assoc(Prot,P2), (G1\ == G2)])}, $\mathcal{C}$ = cause(M2,P2), $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

```
defeasible_rule(cause(M2,P2),[similar(M1,M2),similar(P1,P2),
    have_genotype(G1,M1),have_protein(G1,Prot),
    have_phenotype(G1,P1),cause(M1,Prot),assoc(Prot,P1),
    have_genotype(G2,M2),have_protein(G2,Prot),
    have_phenotype(G2,P2),cause(M2,Prot),assoc(Prot,P2),
    (G1\==G2)])[as(ce_paper01)].
```

- **Different Consistent Explanation**: This argumentation scheme says that *if a group `G1` has a genotype `M`, it has a protein `Prot`, and it has a phenotype `P`, `M` causes `Prot`, `Prot` is associated with `P`, a group `G2` does not have a phenotype `P`, it does not have a genotype `M`, and it does not have the protein `Prot`, then the genotype `M` causes the phenotype `P`.*

  This argumentation scheme has been represented in our approach as: sn = dce_paper01, $\mathcal{P}$ = {have_genotype(G1,M), have_protein(G1,Prot), have_phenotype(G1,P), cause(M,Prot), assoc(Prot,P), ¬have_genotype(G2,M), ¬have_protein(G2,Prot), ¬have_phenotype(G2,P) (G1\ == G2)])}, $\mathcal{C}$ = cause(M,P), $\mathcal{CQ} = \emptyset$.

  Its representation in Jason predicates is given by the defeasible inference rule:

```
defeasible_rule(cause(M,P),[have_genotype(G1,M),
    have_protein(G1,Prot),have_phenotype(G1,P),cause(M,Prot),
    assoc(Prot,P),¬have_genotype(G2,M),¬have_protein(G2,Prot),
    ¬have_phenotype(G2,P), (G1\==G2)])[as(dce_paper01)].
```

Example: As an example for the argumentation schemes presented in [58], the author used a short excerpt, in which they have extracted the following relations and domain knowledge:

$$\Delta = \left\{ \begin{array}{l} \texttt{group(opt)} \\ \texttt{group(affected\_knockout\_mice)} \\ \texttt{have\_phenotype(opt, opt\_pheno)} \\ \texttt{have\_phenotype(affected\_knockout\_mice, ataxia)} \\ \texttt{have\_genotype(opt, Itpr1opt/opt)} \\ \texttt{have\_genotype(affected\_knockout\_mice, Itpr1}\Delta\texttt{18/}\Delta\texttt{18).} \\ \texttt{cause(Itpr1opt/opt, opt\_pheno).} \\ \texttt{similar(opt\_pheno, ataxia)} \\ \texttt{similar(Itpr1opt/opt, Itpr1}\Delta\texttt{18/}\Delta\texttt{18)} \end{array} \right\}$$

In order to test this scenario in our framework, we have represented this knowledge, Δ, in an agent's belief base, as well as we made this agent aware of the argumentation schemes proposed in [58]. Thus, the agent is able to conclude, using our extended argumentation-based reasoning mechanism developed, that cause(Itpr1Δ18/Δ18, ataxia). This result is the same result described in the example from [58].

Argumentation Scheme for Proposed Treatment: The third scenario used to evaluate the generality of our framework comes from the paper [72], in which the authors propose an argumentation scheme used to generate arguments in support of different possible treatment. Providing such arguments to patients, the application aims to help patients to agree about treatment plans. The argumentation scheme for a proposed treatment (ASPT for short) says that *"Given a patient facts* F*, in order to realise the goal* G*, treatment* T *promotes the goal* G*, therefore the treatment* T *should be considered"*. The associated critical question for this scheme are: **CQ1**: Has this treatment been unsuccessfully used on the patient in the past? **CQ2**: Has the patient experienced side effects from this treatment in the past? **CQ3**: Is there an equivalent cheaper treatment for the treatment step of the patient?

This argumentation scheme has been represented in our approach as: sn = aspt, $\mathcal{P}$ = $\{$facts(P, F), goal(P, G), promote(T, G)$\}$, $\mathcal{C}$ = consider(P, T), $\mathcal{CQ}$ = $\{$not(unsuccessfully_used(T, P)), ¬not(experienced_side_effects(P, T)), not(cheaper(T2, T) $\wedge$ promote(T2, G))$\}$.

Its representation in Jason predicates is given by the defeasible inference rule:

```
defeasible_rule(consider(P,T),[facts(P,F), goal(P,G),
          promote(T,G)])[as(aspt)].
```

Further, the critical questions associated to this scheme are represented using the following predicates:

```
          cq(cq1,not(unsuccessfully_used(T,P)))[as(aspt)].
          cq(cq2,not(experienced_side_effects(P,T)))[as(aspt)].
          cq(cq3,not(cheaper(T2,T) & promote(T2,G)))[as(aspt)].
```

Example: The authors in [72] describe a running example in which a patient named Eric has been diagnosed with hypertension. Thus, he desires to reduce hypertension, looking for treatment. The knowledge available to the application is represented in Δ.

$$\Delta = \left\{ \begin{array}{l} \texttt{facts(eric, [age(52), ethnicity(white), weight(overweight)])} \\ \texttt{goal(eric, reduce\_hypertension)} \\ \texttt{promote(tls, reduce\_hypertension)} \\ \texttt{promote(t1l, reduce\_hypertension)} \\ \texttt{promote(t1h, reduce\_hypertension)} \\ \texttt{promote(t2l, reduce\_hypertension)} \\ \texttt{promote(t2h, reduce\_hypertension)} \\ \texttt{cheaper(t1h, t2h)} \\ \texttt{cheaper(t1l, t2l)} \end{array} \right\}$$

According to this scenario, the authors [72] describe that there are 5 different treatments that could be used to reduce Eric's hypertension (the Eric's goal), named `tls, t1l, t1h, t2l` and `t2h`. Considering that `t2l` and `t2h` are more expensive than `t1l` and `t1h`, they do not satisfy the critical question `CQ3`, and therefore there is no acceptable instance of argument for those treatments. Thus, only the treatments `tls, t1l` and `t1h` are acceptable instances of this argumentation scheme. `tls, t1l` and `t1h` are proposed as treatment options. Our argumentation-based reasoning mechanism reach the same result for the acceptable treatments described in the example from [72].

4.5.2 Argumentation Schemes for Data Access Control

The last argumentation schemes considered in our evaluation have been proposed in [100], in which we have analysed the problem of data access control between multi-agent systems. In this particular work, we proposed to solve that problem using argumentation-based interface agents. The interface agents make decisions based on the proposed argumentation schemes for data access control and the argumentation-based reasoning mechanism described in the previous sections. In this section, we present the argumentation schemes for data access control and our research describing how we reach the modelling of those reasoning patterns.

Our motivation is that security concerns are an increasingly important issue, mainly regarding sharing and using data, given the increasing use of IoT (*Internet of Things*) devices and the proliferation of big data mechanisms [71]. IoT has emerged from the idea that everything can be connected, adding new dimensions to the world of information and communication technology [40]. Given the vision of IoT, the number of devices connected to the *Internet* has been increasing and consequently the amount of data available on the internet has also been increasing [141]. This opens opportunities for studies in areas like big data analysis [166], machine learning [118], and many others.

Data access/sharing control were not sufficiently studied in previous decades, considering that data were shared within boundaries of organisations, e.g., companies and universities. In such organisations, trust and security issues were easily solved [71]. However, with the recent proposal of IoT, including the integration of different smart applications (e.g., healthcare, smart cities, smart home/building, etc.), sharing information between different systems has become mandatory [118]. Consequently, the problems of data sharing/access control and privacy protection have become challenging issues when integrating different smart applications [168].

System of Systems (SoS) [159] is a natural way to think of the modelling of these current ideas from the integration of different smart applications, in which heterogeneous systems (smart applications) are modelled as subsystems, which cooperate to achieve a higher purpose of the whole system (e.g., a global IoT). Multi-Agent Systems (MAS) [164] provide an interesting paradigm to implement SoS models; in particular, MAS provides a suitable approach to implement smart applications in IoT[7] [56]. Thus, it is a natural choice to think of the modelling of such systems using multi-agent systems, although data access control in such systems seems not sufficiently studied.

Here, we are interested in the communication interface that implements data access control between different smart applications, which is currently a challenge in the integration of different systems, mainly because of the uncertainty contained in the information used during this decision making process [71, 69]. In that respect, we propose the modelling of data access control interfaces using argumentation-based agents. Also, we propose two argumentation schemes (i.e., reasoning patterns) for data access control. These reasoning patterns take into consideration the most relevant models for data access control in the literature [11, 48, 4], generalising the reasoning an agent needs to carry out when treating a request. Guided by the critical questions in the schemes, agents are able to deal with the uncertainty about the information used during this decision making process. Furthermore, considering our approach for data access interface using argumentation-based agents, agents are able to understand why a request has been denied to it. Understanding this answer allows agents to provide additional information in order to be correctly categorised, as well as to expose emergency situation in which emergency access control rules may apply. The argumentation schemes we will present in this section have been published in [100].

Access Control Models

Role-Based Access Control (RBAC) models provide mechanisms to protect resources from unauthorised use in an organisation. In such models, instead of specifying all the actions (accesses) each user is allowed to execute, each attempt to access an object is specified by categorising users into *roles* [3]. Thus, RBAC models assume that different individuals are classified into roles[8] and

[7]We refer the reader to IoA (*Internet of Agents*) workshop series for an overview about this topic http://ioa.alqithami.com/.
[8]Similarly to roles present in multi-agent systems based on the organisational paradigm.

different roles have different access permissions to data [11, 48]. However, in RBAC models, role is the only form of category, while more general approaches can be defined. Consequently, in [11], the author proposes a meta-model for access control based on categories, arguing that having a common/agreed semantics is essential when access control information needs to be shared.

In Category-Based Access Control (CBAC) meta-model [11], a category[9] is any class or group to which entities are designated [11, 4], and entities are constants from the domain of discourse. Following [11, 4], a CBAC model is defined by a countable set $\mathcal{C}$ of categories, denoted as $\{c_0, c_1, \dots, c_n\}$; a countable set $\mathcal{P}$ of principals (the entities that are able to require access to resources), denoted $\{p_0, p_1, \dots, p_n\}$; a countable set $\mathcal{A}$ of actions, denoted $\{a_0, a_1, \dots, a_n\}$; a countable set $\mathcal{R}$ of resources identifiers, denoted $\{r_0, r_1, \dots, r_n\}$; a finite set $Auth$ of possible *answers* to access request $\{\texttt{grant}, \texttt{deny}, \texttt{undetermined}\}$, and a set $\mathcal{S}$ of *situations identifiers* to denote environment information, which are application dependent [11], and they can represent, for example, IP address, time instants, system state, external state, etc.

The following relations are defined for a CBAC meta-model [11]: (i) *principal-category assignment*: $\mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C}$, such that $(p, c) \in \mathcal{PCA}$ iff a principal $p \in \mathcal{P}$ is assigned to category $c \in \mathcal{C}$; (ii) *permission-category assignment:* $\mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$, such that $(a, r, c) \in \mathcal{ARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ can be performed by principals assigned to category $c \in \mathcal{C}$; (iii) *authorisations*: $\mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{PAR}$ iff a principal $p \in \mathcal{P}$ can perform action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$. Thus, a principal p is authorised to perform the action a on a resource r only if p belongs to a category c such that for some category below c in the hierarchy (and including c itself) the action a is authorised on r, otherwise the request is `denied`. The general idea is summarised by the following rule, in which `subc(c',c)` checks if `c'` is a subcategory of `c`:

$$(p, a, r) \in \mathcal{PAR} \Leftarrow (p, c) \in \mathcal{PCA} \wedge subc(c', c) \wedge (a, r, c') \in \mathcal{ARCA}$$

In [4], the authors extend the CBAC model in order to consider policy composition, where an access control policy is combined with an emergency policy that specifies how various emergency situations affect the rights of users to access resources. Thus, two access control policies are considered, π_1 and π_2. π_1 describes the usual access control rules, and π_2 the emergency policy, describing emergency situations `s` in which the access may be granted for a particular category of principals `c`.

$$(p, a, r) \in \mathcal{PAR} \Leftarrow (p, c) \in \mathcal{PCA} \wedge \texttt{subc(c}', \texttt{c)} \wedge (a, r, c') \in \mathcal{ARCA}_{\pi_1}$$

$$(p, a, r) \in \mathcal{PAR} \Leftarrow (p, c) \in \mathcal{PCA} \wedge \texttt{subc(c, c}'\texttt{)} \wedge \texttt{emrg(s)} \wedge (a, r, c') \in \mathcal{ARCA}_{\pi_2}$$

[9]Here, categories are defined as the classification types used in access control [11], differently from [59].

As an example, [4] describes a hospital scenario, in which, although only family doctors may have access to the record of their patients, during an emergency situation, all doctors (i.e., a super-category) may have access to the record of that patient.

$$(\mathtt{D}, \mathtt{read}, \mathtt{record(P)}) \in \mathcal{PAR} \Leftarrow (\mathtt{D}, \mathtt{doctor}) \in \mathcal{PCA} \ \wedge \ \mathtt{emrg(cardiac\_emrg, P)} \wedge (\mathtt{read}, \mathtt{record(P)}, \mathtt{doctor}) \in \mathcal{ARCA}_{\pi_2}$$

An Architecture for Data Access Control

Based on the access control models presented, here we propose an architecture for data access control, considering the modelling of a system based on SoS and MAS. In other words, we propose an approach for data access control in which the whole system is composed of subsystems (SoS) and each subsystem is implemented as a MAS. First, we describe how we define the access control policy, considering not only the usual access control rules from [11], but also considering the idea of emergency access control rules from [4]. Afterwards, we describe how we structured the hierarchies of categories for information and agents, using not only the idea of categories from [11], but also categories of information, which allows us to use a category to describe the information. Finally, we describe how external requests are treated by a subsystem, considering the access control policy and the external information available to agents dealing with the request.

Access Control Policy

An access control policy is defined by a set of access control rules, which are specified according to the application needs. One of the most natural ways to define an access control policy is specifying it during the project of the system [2]. Another option is to start with a predefined access control policy and dynamically adjust it according to the system needs. For simplicity, we will consider that the access control policy is defined during the system design phase; however, note that a dynamic access control policy could be used in our approach, given that agents have an explicit representation of the rules and could use the updated access control policy when making decisions.

Access control rules specify which categories of agents have access to which categories of information. Thus, we use two distinct kinds of categories in this work. The first, *access-category*, is the usual concept for categories in CBAC [11], describing categories of principals, here agents, which request to access resources, here information. The second, *information-category*, is used for categories of information. Using information-category not only allows us to group information of similar privacy but also allows us to describe hierarchies between categories of information. Both characteristics provide a semantic description of the information belonging to each category, which is essential when applying argumentation-based techniques in such domains.

Besides the access control rules from CBAC [11], we also consider emergency access control rules from [4] as part of the access control policy. Thus, the resulting access control rules have the following format:

$$\begin{aligned}
&(\textit{access-category}(r_i) && \xrightarrow{\text{access}} \textit{information-category}(c_i))\\
&(\textit{access-category}(r_i) \wedge \textit{emergency}(s_i) && \xrightarrow{\text{access}} \textit{information-category}(c_i)).
\end{aligned}$$

meaning that “the access-category r_i has access to the information-category c_i” and “during an emergency situation s_i, the access-category r_i has access to the information-category c_i”, respectively.

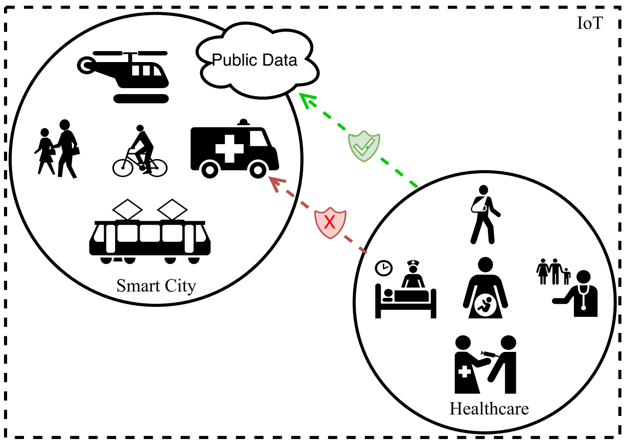
While the assignment of information to an information-category and the access control rules are internally defined in the multi-agent system that owns the information using only information from the system itself, external requests will require additional information from the requester in order to categorise it according to the local definition of access-category and, consequently, to support decisions on whether to grant access to the information or not. Each category in access-category is defined by a set of constraints that the requester needs to satisfy in order to be so categorised. Constraints can include different characteristics of the requester, i.e., its role and reliability, as well as characteristics of its environment, organisation, domain, and so forth [11].

A Structure for Categories

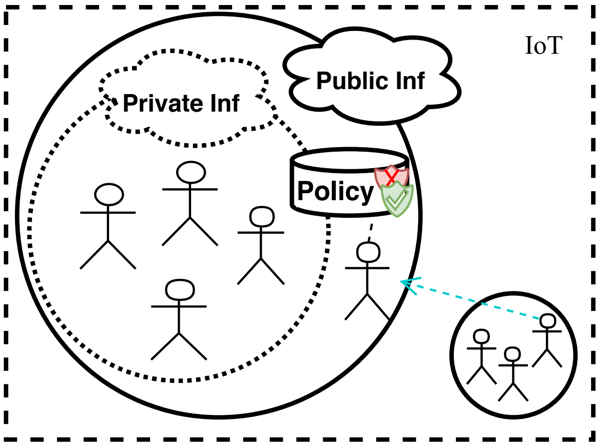
When considering different smart applications, within the IoT domain for example, we are able to think of the information generated by each application as being categorised into two major categories, *public* and *private* information. Public information is the contribution that a subsystem makes to the whole system, resulting from the processing of other information in order to avoid disseminating private information, e.g., about the end-users and/or the system. Figure 4.1(a) illustrates this idea, in which *public* data are available for external requests, but *private* data, e.g., the location of a vehicle in a smart city, are not available for external requests. In summary, all applications might have access to public data, and access control policies may have an access control rule that provides access to *public* information for all agents in the system. Differently, *private* information may be granted access only to an access-category for which the access to such information is specified in the access control policy, i.e., there is an access control rule granting access for the access-category of the requester to the information-category of that information.

In Figure 4.2(a), we show the hierarchy between the different information-category. We consider that information is either *private* or *public*. Also, private information is categorised as end-user and system information. Note that an access control rule that grants access for an access-category to the information-category *private* also grants access to the information-category *end-user* and *system*.

In Figure 4.2(b), we show how we consider the hierarchy between the different access-category. At the root, we have a general access-category of all agents, which includes both subcategories of *internal* and *external* agents, representing agents playing some role inside that subsystem, and external agents which do not play any role in that subsystem. For both *internal* and *external* agents, a number of subcategories can be defined, based on the different roles considered and

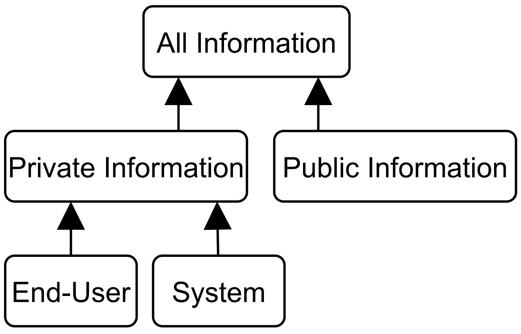


(a) Data Access Between Different Application.

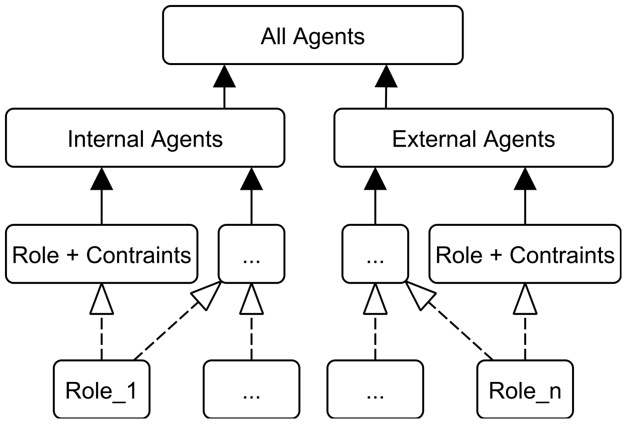


(b) Interface Agents for Data Access Control.

Figure 4.1 – An Architecture for Data Access Control.



(a) *Information-Category* Hierarchy.



(b) *Access-Category* Hierarchy.

Figure 4.2 – A Structure for Categories in Data Access Control.

other constraints required for each category in that system. Thus, an agent can be categorised in more than one access-category, and two agents playing the same role may not be categorised in the same access-category, given that the role is only part of the definition of the categories. Note that a subcategory inherits the access from superior categories, so if there is a rule granting access for the access-category *all agents* to the information-category *public information*, that means it grants access for access-categories *external agents*, *public information*, and any subcategory of those too.

Interface Agents for Data Access Control

In SoS, heterogeneous systems cooperate to achieve a higher purpose [159]. During the development of SoS, the communication interfaces are considered one of the more complex and difficult tasks [159]. When implementing SoS using the MAS paradigm, the communication interface with external systems is implemented using specialised agents. Thus, a communication interface for data access control corresponds to agents which are responsible for making a decision about sharing or not the information according to who is the requester and the access control policy of that particular system. Figure 4.1(b) illustrates an interface agents for data access control.

When the agent responsible for the communication interface receives a request from an external agent a_1 to access information i_1, it carries out an argumentation-based reasoning process in order to construct an acceptable argument that grants access for a_1 to i_1, considering the access control policy (the information-category of i_1, the access control rules, and the access-category constraints). When this agent is able to construct an acceptable argument granting access for requester a_1 to information i_1, that information access request is granted, otherwise it is denied.

Argumentation Scheme for Data Access Control

Based on the meta-model introduced in [11], as well as the emergency policies introduced in [4], we introduce an Argumentation Scheme for Data Access Control named AS4DAC. The argumentation schemes presented in this section have been published in [100].

[premise] Information I has security information-category C. **[premise]** Agent A belongs to an access-category R which has access to information with security information-category C. **[conclusion]** Agent A has access to information I.

This conclusion is reached unless the answer to any of the following questions is *no*:

CQ1 Does information I belong to information-category C?

CQ2 Does agent A belong to access-category R?

CQ3 There exists an access control rule that grants access for R to I?

CQ4 Is this conclusion free from conflict with any other information-category C_i to which information I is also assigned, which is not a super-category of C, and where there is no access control rule that grants access for R to C_i?

CQ5 Is this conclusion free from conflict with any other access-category R_i that the agent A is also allocated to, which is not a super-category of R, and where there is no access control rule that grants access for R_i to C?

CQ6 In the case where this conclusion is based on an emergency access control rule that grants access for R to C during an emergency situation S_i, is S_i the case?

As mentioned above, while the attribution of information to information-category and the access control rules (the access control policy) are internally defined in the multi-agent system which owns the information, external requests will require additional information from the requester in order to categorise it according to the internal definition of the access-category and, consequently, to make a decision about granting access to that information or not. Considering the defeasibility of this information, it is necessary to give special attention to this process when receiving a request. Therefore, we introduce the Argumentation Schemes for Access-Category Attribution, named AS4ACA, which allows agents to investigate the attribution of a requester to access-category in more depth.

> **[premise]** An access-category R is defined by a set of constraints S. **[premise]** Agent A satisfies the constraints S. **[conclusion]** Agent A belongs to the access-category R.

CQ1 Does agent A satisfy all constraints s_i in the set of constraints S?

CQ2 Is R the more specific access-category for which A satisfies the constraints?

Note that not only should an agent satisfy all constraints required by a particular access-category in order to be so categorised, but it should be also categorised in the more specific access-category it satisfies the constraints, given more specific categories inherit access from the super-categories in the hierarchy of access-categories.

Representing the Argumentation Schemes

Considering the argumentation schemes introduced in the previous section, `AS4DAC` can be formalised in our framework as follows: $\mathcal{SN}$ = `AS4DAC`; $\mathcal{C}$ = `access(A,I)`; $\mathcal{P}$ = {`inf_category(I,C)`, `ac_category(A,R)`, `access(R,C)`}. The critical questions are represented as follows:

- CQ_1 = { `inf_category(I,C)` };
- CQ_2 = { `ac_category(A,R)` };
- CQ_3 = { `access(R,C)` };
- CQ_4 = { `not( inf_category(I,C2), ¬subc(C,C2), ¬access(R,C2)`)};
- CQ_5 = { `not( ac_category(A,R2), ¬subc(R,R2), ¬access(R2,C)`)};
- CQ_6 = { `not( emrg(S,access(R,C)), ¬S`)}

Also, `AS4ACA` can be formalised as follows: $\mathcal{SN}$ = `AS4ACA`; $\mathcal{C}$ = `ac_category(A,R)`; $\mathcal{P}$ = `constr(S,R), satisfies(A,S)`. The critical questions are represented as follows:

- CQ_1 = { `in(Si,S),constr(S,R), satisfies(A,Si)` };
- CQ_2 = { `not( subc(R2,R),constr(S2,R2), satisfies(A,S2)`)};

Note that the conclusion of AS4ACA, i.e., ac_category(A,R), is one of the premises in AS4DAC. When an agent answers *CQ₂* in AS4DAC, i.e., it checks the access-category of the requester, an instance of the argumentation scheme AS4ACA is used to infer that information, bringing to light the particular critical questions from that scheme, i.e., CQ_1 and CQ_2 in AS4ACA. This is an example of the use of nested argumentation schemes, which is an underdeveloped area of study in the argumentation literature.

As mentioned above, the attribution of agents to access-category is made by the systems receiving the request, even with the information coming from external sources. Thus, it makes sense to use a specific argumentation scheme in order to guide the verification of the truth of this external information. In contrast, the attribution of information to information-category and the definition of the access control policy are made internally by the multi-agent system which receives the request, thus more simple/direct verification is possible and the critical questions in AS4DAC account for that.

Argumentation-Based Reasoning for Data Access Control

An example of argument, instantiated from AS4DAC, is given below, with $\theta = \{\mathtt{I} \mapsto \mathtt{i_1}, \mathtt{C} \mapsto \mathtt{c_1}, \mathtt{A} \mapsto \mathtt{a_1}, \mathtt{R} \mapsto \mathtt{r_1}\}$:

$\langle$ {inf_category(i_1,c_1), ac_category(a_1,r_1), access(r_1,c_1),
  [inf_category(I,C), ac_category(A,R), access(R,C) $\Rightarrow$ access(A,I)]},
  access(a_1,i_1) $\rangle$

Considering our approach for data access control, an agent may grant access for a requester a_1 to information i_1 only if there is an ***acceptable*** argument concluding access(a_1, i_1), considering the existence of an access control rule that grants access for the access-category of a_1 to the information-category of i_1. Remembering that an instance of argument $\langle S, c\rangle_{sn}^{\theta}$ from an argumentation scheme $\mathcal{SN}$, is ***acceptable*** to an agent ag (where Δ_{ag} is its knowledge base) if: (i) all premises in S are acceptable to ag, i.e., $\forall p\theta \in \mathcal{S}, \Delta_{ag} \models p\theta$, either because p is asserted in its knowledge base, or because p is the conclusion of an acceptable argument; and (ii) all critical questions related to the argumentation scheme $\langle \mathcal{SN}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$ are positively answered by ag, i.e., $\forall Cq_i \in \mathcal{CQ}, \Delta_{ag} \models Cq_i\theta$.

Considering the example of argument above, this argument is acceptable to an agent ag when: (i) $\Delta_{ag} \models \mathtt{inf\_category}(i_1, c_1)$, which is asserted in Δ_{ag}; (ii) $\Delta_{ag} \models \mathtt{ac\_category}(a_1, r_1)$, which requires ag to instantiate an acceptable argument from the argumentation scheme AS4ACA; (iii) $\Delta_{ag} \models \mathtt{access}(r_1, c_1)$, which is asserted in Δ_{ag}; and (iv) all critical questions are also positively answered by ag: $\Delta_{ag} \models \mathtt{inf\_category}(i_1, c_1)$, $\Delta_{ag} \models \mathtt{ac\_category}(a_1, r_1)$, $\Delta_{ag} \models \mathtt{access}(r_1, c_1)$, $\Delta_{ag} \models \{\mathtt{not}(\mathtt{inf\_category}(i_1, c_2), \neg\mathtt{subc}(c_1, c_2), \neg\mathtt{access}(r_1, c_2))\}$, $\Delta_{ag} \models \{\mathtt{not}(\mathtt{ac\_category}(a_1, r_2), \neg\mathtt{subc}(r_1, r_2), \neg\mathtt{access}(r_2, c_1))\}$, $\Delta_{ag} \models \{\mathtt{not}(\mathtt{emrg}(s_1, \mathtt{access}(r_1, c_1)), \neg s_1))\}$.

Note that an agent will always be able to categorise an information i_1 to an information-category and a requester a_1 to an access-category[10], given the hierarchy of categories defined in Figure 4.2. Thus, when an access to i_1 is denied, i.e., the agent is not able to construct an acceptable argument for access(a_1, i_1), that means (i) there is no access control rule granting access for the access-category of a_1 to the information-category of i_1 (**CQ3** in AS4DAC); (ii) there exist a counter-example for access(a_1, i_1) (**CQ4** and **CQ5** in AS4DAC); or, (iii) the emergency situation considered, if any, is not true (**CQ6** in AS4DAC). In all cases, the agent denies access for a_1 to i_1 with the following argument:

$$\langle\ \{\texttt{inf\_category}(i_1,c_1),\ \texttt{ac\_category}(a_1,r_1),\ \neg\texttt{access}(r_1,c_1),$$
$$[\texttt{inf\_category}(I,C),\ \texttt{ac\_category}(A,R),\ \neg\texttt{access}(R,C)\ \Rightarrow\ \neg\texttt{access}(A,I)]\},$$
$$\neg\texttt{access}(a_1,i_1)\ \rangle$$

When an agent a_1 receives this argument, it is able to respond with additional information. This information may clarify to the agent dealing with the request the correct access-category to categorise a_i, answering differently **CQ2** in AS4DAC (i.e., the conclusion of AS4CAC) and, possibly, answering positively **CQ3**, **CQ4** and **CQ5** in AS4DAC. Also, a_1 may provide information about an emergency situation that may grant access to this information, considering an emergency access control rule.

Scenarios of Data Access Control

Travelling Patient

In our first scenario, we illustrate an emergency situation that requires sharing data between two different hospitals. In our scenario, a patient named *Bob* is being monitored and receiving treatment in a hospital **H1** located in Brazil. *Bob* decides to travel to the UK, receiving a bracelet that informs how to contact **H1** in case of emergency. During the time *Bob* is staying in the UK, *Bob* starts to feel a strong pain in the chest, becomes unconscious, and goes to hospital **H2** located in the UK.

As *Bob* is unconscious (it seems an emergency situation), a receptionist of **H2**, named *Anna*, calls **H1** in order to know the patient record.

$$\pi_{\mathbf{H1}} = \left\{ \begin{array}{ll} \textit{a-c(all)} & \xRightarrow{\text{access}} \textit{i-c(public)} \\ \textit{a-c(internal-family\_doctor)} & \xRightarrow{\text{access}} \textit{i-c(private-end\_user-patient\_data)} \\ \textit{a-c(internal-doctor)} \wedge \textit{emrg(unc)} & \xRightarrow{\text{access}} \textit{i-c(private-end\_user-patient\_data)} \\ \textit{a-c(external-hospital\_doctor)} \wedge \textit{emrg(unc)} & \xRightarrow{\text{access}} \textit{i-c(private-end\_user-patient\_data)} \end{array} \right\}$$

[10]However, this categorisation may change as the agent acquires more information about a_1.

Considering the access control rules[11] of **H1**, specified in $\pi_{\mathbf{H1}}$, the agent responsible for answering the request categorises *Anna* to the access-category *external-hospital_receptionist*. Thus, it is not able to construct an acceptable argument for `access(anna, record(bob,R))`, denying access to that information. As *Anna* has been correctly categorised, she sends a message ⟨**H1**, `consider`, `emrg(is(bob,unc))`⟩, asking to **H1** to consider the emergency situation in which *Bob* is unconscious; however, the information is denied again.

Thus, *Anna* passes the phone call to a doctor named *John* who is treating that patient at **H2**, and *John* requests the records. The agent responsible for dealing with the request categorises *John* to the access-category *external-hospital_doctor*, and provides the information to *John*, given they already know the emergency situation `emrg(is(bob, unc))`, and *external-hospital_doctor* has access to *private-end_user-patient_data* during such emergency situations.

Smart Building

The second scenario illustrates an emergency situation (fire) in a smart building **B1**. During the rush time, when many people are in the building, an emergency situation due to fire is detected and the building is evacuated. **B1** is able to detect how many people are inside the building, which is public information; however, the location of each person inside the building is available only for the *internal-manager* and for an *external-fireman* in emergency fire situations, i.e., `emrg(fire)`. When a fireman arrives, it requests access to information on how many people are inside the building, which is public information, thus **B1** answers that there are 2 people inside the building. With this information, the *fireman* requests the location of each person inside the building.

$$\pi_{\mathbf{B1}} = \left\{ \begin{array}{lll} \textit{a-c(all)} & \xRightarrow{\text{access}} & \textit{i-c(public)} \\ \textit{a-c(internal-manager)} & \xRightarrow{\text{access}} & \textit{i-c(private-end\_user)} \\ \textit{a-c(external-fireman)} \wedge \textit{emrg(fire)} & \xRightarrow{\text{access}} & \textit{i-c(private-end\_user)} \end{array} \right\}$$

Considering the access control rules of **B1**, specified in $\pi_{\mathbf{B1}}$, and the already known emergency fire situation, i.e., `emrg(fire)`, **B1** provides the information to the fireman.

4.5.3 Evaluating the implementation

We have evaluated our implementation of the extended argumentation-based reasoning mechanism considering argumentation schemes in two main points. The first point was to have an implementation that returns the same result from the scenario extracted from the literature and implemented in our framework. That is, we evaluate our implementation, representing the

[11]We are using a-c for access-category and i-c for information-category.

argumentation schemes from the previous sections, and analysing whether our implementation allows agents to reach the same conclusion described in the examples of each argumentation scheme from the literature. For all examples, our implementation reaches the expected result.

The second point was to evaluate the influence of the number of premises and critical questions in the time necessary to agents reach a conclusion in our approach. This evaluation is important because it can guide future developers on how to model argumentation schemes for particular applications, e.g., the argumentation scheme for data access control we modelled.

In order to evaluate the time required to reach a conclusion, we have generated argumentation schemes, varying the number of critical questions and premises. The results are shown in Figure 4.3 and 4.4

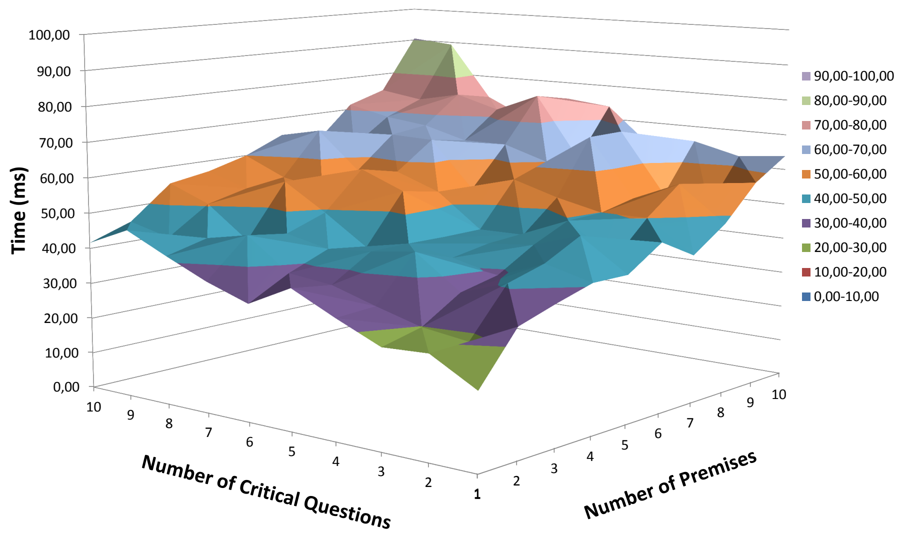


Figure 4.3 – Time (ms) for reaching a conclusion considering one level of inference.

In the first experiment, we have fixed only one level of inference for argumentation schemes, that means that premises and critical questions are directly asserted in agents' belief base. Thus, an agent need only to find an argumentation scheme that allows it to instantiate an acceptable instance of argument concluding that particular information.

Figure 4.3 shows the results of our experiments, varying the number of premises and critical questions from 1 to 10. It can be noted that the influence of checking the premises and critical questions in the final time to reach that particular conclusion is similar.

When considering 2 levels of inference, that means, premises of an argumentation scheme are the conclusions of another argumentation scheme in which premises are asserted in the agents' belief bases, we obtain a different result.

Figure 4.4 shows the result of our experiments, varying the number of premises (and respective argumentation schemes, given each premise is the conclusion of another argumentation scheme) and the critical questions. It can be noted that the number of premises has a greater influence on the time to reach a particular conclusion than the number of critical questions. This is resulting from the fact that premises of the argumentation scheme used to infer that particular conclusion are the conclusion of another argumentation scheme with its own critical questions, which also need to be positively answered in order to be acceptable.

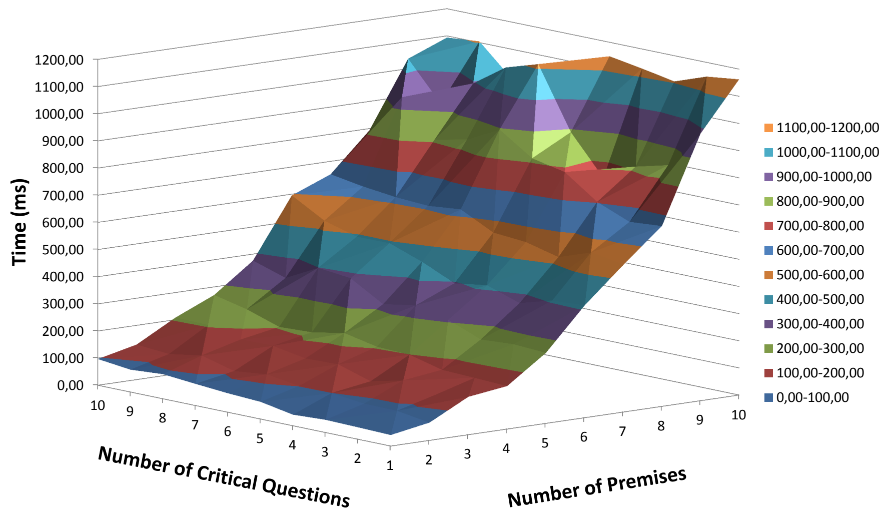


Figure 4.4 – Time (ms) for reaching a conclusion considering 2 levels of inference.

4.6 Final Remarks

In this chapter, we presented an extended version for our argumentation-based reasoning mechanism we have developed in an agent-oriented programming language, published in [101, 109], in order to consider argumentation schemes. We have evaluated our approach by taking some argumentation schemes from different areas in the literature, and representing those scheme in our framework. Also, in order to demonstrate the process of modelling an argumentation scheme for a particular application domain, we have proposed argumentation schemes for data access control. Using the argumentation schemes for data access control and our extended argumentation-based reasoning mechanism, we are able to implement argumentation-based interface agents that implement data access control in multi-agent applications.

Our approach for data access control was built considering: (i) the models for data access control from [11, 48, 4, 62] and work that applied such models in the specification of access control policies [3, 2, 1]; (ii) approaches that apply argumentation-based techniques considering the specification of argumentation schemes in their conception [143, 89, 113]; (iii) the problem of data access control [71, 168, 37, 161, 75, 160, 135, 138]; and (iv) our argumentation-based framework for argumentation schemes in multi-agent systems.

To the best of our knowledge, our work is the first to propose argumentation schemes for data access control. We specified those reasoning patterns considering characteristics of the most relevant models for data access control, providing a general approach for reasoning about data access control. Others approaches have modelled problems of data access control/sharing using argumentation. In [71, 69, 70], the authors model the problem of data access control/sharing using argument rules and define preferences between such rules, thus they are able to treat the problem as an argumentation problem, solving the conflicts present in the access control policy. Note that our work is more general than those, given that we proposed a reasoning pattern that agents use during reasoning and decision-making, considering the access control policy of that system. Thus, instead of translating the access control policy to an argumentation problem, in our approach, an agent carries out a decision-making process using argumentation-based reasoning, guided by our schemes, considering the access control policy and the relevant external information about the request. In [119], the authors propose an argumentation-based protocol in which a client agent may persuade a server agent in order to gain access to information. The main idea in that work is that the client's arguments could persuade the server agent, making it change the permission previously denied for that client. Our work differs from [119] given that, in the context of our work, an external agent should not be able to change the internal system's access control policy. An evolutionary access control policy may result from argumentation-based dialogues within the system, in order to adapt its access control policy, which is part of our intended future work.

Finally, we have evaluated the time required by agents to reach a conclusion using our framework for argumentation schemes, depending on the number of premises and critical questions in each scheme, and the level of depth considered. In our experiments, we considered only two levels of depth, given most of the approaches from the literature consider only one argumentation scheme in their application. Our work [100] seems to be one of the first work to consider nested argumentation schemes (2 levels of depth).

5. ARGUMENTATION-BASED DIALOGUES IN AGENT-ORIENTED PROGRAMMING LANGUAGES

In the previous chapter, we described the argumentation-based reasoning mechanism we develop in an agent-oriented programming language, and how we extended it considering argumentation schemes. Using the argumentation-based reasoning mechanism, agents are able to reach well-supported conclusions/beliefs. Also, agents are able to reason about new information perceived from the environment or received from other agents during communication. An agent needs only to include such perceived or received information in its belief base (with the appropriate source of information) and query if that is acceptable for it, given the current belief base state[1]. Besides improving their reasoning capabilities, agents are able to rationally engage themselves in argumentation-based dialogues, reasoning about the arguments received from other agents, evaluating and generating arguments to position themselves in dialogues.

Agent communication normally is based on speech acts theory [137], where different speech acts have a different meaning, even when the content used is the same. The literature of argumentation-based dialogues follows the same principles, proposing performatives to enable argumentation-based dialogues [8, 7, 114, 116, 81], which are regulated by some rules defined through protocols. However, such work does not focus on the effects of such speech act in the internal state of agents, as [151], which could formally define the communication between agents in argumentation-based dialogues towards proving important properties and guaranteeing precision and coherence in such communications.

Towards argumentation-based dialogues in agent-oriented programming languages, we have identified the most common speech acts in the literature mentioned above, and we have formalised such performatives using operational semantics [120] in BDI-based agent-oriented programming languages (e.g., Jason). The operational semantics provide us with precise effects of communications. Its implementation combined with the argumentation-based reasoning mechanism, described in Chapter 4, and a protocol enable argumentation-based dialogues in multi-agent systems.

Therefore, in Section 5.1, we start this chapter describing the speech acts we selected from the literature and its operational semantics in BDI-based agent-oriented programming languages, previously published in [111, 108]. In section 5.2, we briefly describe a CArtAgO artifact we developed to support argumentation-based dialogues, which implements the commitment stores formalised in our semantics, mentioning the protocol for task reallocation published in [110]. Also, in Section 5.2, we describe a Jason module we have implemented, which facilitates the implementation of different protocols in Jason agents. After that, in Section 5.4 we briefly describe how argumentation schemes can be considered in argumentation-based dialogues. After that, in Section 5.5, we evaluate our framework for argumentation scheme in multi-agent systems in the

[1]Note that our approach guarantees that the acceptability of arguments is verified using the updated agents‘ belief base, i.e., they consider their last and updated mental state when querying the acceptability of arguments.

dialogue perspective. In order to evaluate our framework, we model two different protocols, and we show different dialogues resulting from those protocols, based on different configurations of agents. Finally, in Section 5.6, we describe some computational benefits our approach brings to multi-agent system communication, in which agents can use enthymemes (incomplete arguments) instead of complete arguments. Our approach for enthymemes in multi-agent system brings benefits related to agents' rationality and economy, in which agents communicate only the essential information to understand each other.

5.1 Formal Semantics

5.1.1 New Performatives for AgentSpeak

The performatives selected to enable argumentation-based dialogues in AgentSpeak are presented below[2], along with the intended (informal) meaning:

- `assert`: an agent that sends an `assert` message 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. As described in [162], the assert performative could be used both to express an entire argument within a single locution or to express, individually or in combination, the constituent parts of an argument (conclusion and premise(s)).
- `accept`: an agent that sends an `accept` message declares, to all participants of the dialogue, that it accepts a previous claim of another agent. The receivers of the message become aware of this acceptance.
- `retract`: an agent that sends an `retract` message declares, to all participants of the dialogue, that it is no longer committed to defending its previous claim. The receivers of the message become aware of this fact.
- `question`: an agent that sends an `question` message declares, to all participants of the dialogue, that it desires to know the reasons for a particular piece of information. The receiver of the message presumably will provide the support set (an argument) for that information.
- `challenge`: the `challenge` performative is similar to `question`, except that the sender of the message is committed to defending a claim contrary to the previous claim of another agent. It requires a previous assert or justify message in which that claim has been introduced by another agent.

[2]These performatives and their formal semantics have been published in [111, 108] and they are described here for completeness. Later in this paper, we extend this semantic in order to accommodate argumentation schemes, the main focus of our research.

- `justify`: the `justify` is used to respond to a `question` or `challenge` message[3].

Further, performatives `opendialogue` and `closedialogue` are used for creating and concluding dialogues, respectively; and two other performatives, `acceptdialogue` and `refusedialogue`, are used by the participants to accept or refuse taking part in a dialogue, respectively.

5.1.2 The Basis for the Operational Semantics

We define the semantics of speech acts for argumentation-based dialogues in AgentSpeak using operational semantics, a widely used method for giving semantics to programming languages [120]. We use a multi-level semantics representation we introduced in [104, 105]. The operational semantics is given by a set of inference rules that define a transition relation between configurations $\langle AG, D\rangle$ of the multi-agent system[4] where:

- The AG component is a set of tuples $\langle id, Conf\rangle$ representing each agent in the society, where each agent is identified by a unique identifier id and the agent current internal state is represented by $Conf$. The agent state is in fact given by a *configuration* of the operational semantics of AgentSpeak as formalised in the existing literature (e.g., [151]); we assume some familiarity with the semantics of AgentSpeak.
- The set of all dialogues in that society, D, is a set of tuples $\langle did, Ags, Status\rangle$ where:
  - did is a dialogue identifier (which is unique for each dialogue within that multi-agent system);
  - Ags is a set of tuples $\langle id, CS\rangle$, where id identifies a particular agent that is participating in the dialogue and CS is its *commitment store*;
  - $Status$ represents the status of the dialogue and for the time being we assume it is one of only two values: `OPEN` if the dialogue is ongoing and `CLOSED` otherwise.

The agent configuration ($Conf$) is given by a tuple $\langle ag, C, M, T, s\rangle$, originally defined in [151], where:

- ag is a set of beliefs bs and a set of plans ps.
- An agent's circumstance C is a tuple $\langle I, E, A\rangle$ where:

[3]While some protocols suggest that `assert` messages could be used to respond a `question` or `challenge` messages, we think that using the `justify` helps programmers to design and implement protocols.

[4]We use only components that are needed to demonstrate the semantics, but we emphasise the existence of other components such as roles, norms, etc.

  - I is a set of *intentions* $\{i, i', \ldots\}$. Each intention i is a stack of partially instantiated plans.
  - E is a set of *events* $\{(te, i), (te', i'), \ldots\}$. Each event is a pair (te, i), where te is a triggering event and i is an intention — a stack of plans in case of an internal event, or the empty intention T in case of an external event. For example, when the belief revision function (which is not part of the AgentSpeak interpreter but rather of the agent's overall architecture), updates the belief base, the associated events — i.e., additions and deletions of beliefs — are included in this set. These are called *external* events; internal events are generated by additions or deletions of goals from plans currently executing.
  - A is a set of *actions* to be performed in the environment.

- M is a tuple $\langle In, Out, SI \rangle$ whose components characterise the following aspects of communicating agents (note that communication is typically asynchronous):

  - In is the mail inbox: the multi-agent system runtime infrastructure includes all messages addressed to this agent in this set. Elements of this set have the form $\langle mid, id, ilf, cnt \rangle$, where mid is a message identifier, id identifies the sender of the message, ilf is the illocutionary force of the message, and cnt its content: a (possibly singleton) set of AgentSpeak predicates or plans, depending on the illocutionary force of the message.
  - Out is where the agent posts messages it wishes to send; it is assumed that some underlying communication infrastructure handles the delivery of such messages. Messages in this set have exactly the same format as above, except that here id refers to the agent to which the message is to be sent.
  - SI is used to keep track of intentions that were suspended due to the processing of communication messages; the intuition is as follows: intentions associated with illocutionary forces that require a reply from the interlocutor are suspended, and they are only resumed when such reply has been received.

- When giving semantics to an AgentSpeak agent's reasoning cycle, it is useful to have a structure which keeps track of temporary information that may be subsequently required within a reasoning cycle. T is a tuple $\langle R, Ap, \iota, \varepsilon, \rho \rangle$ with such temporary information; these components are as follows:

  - R is the set of *relevant plans* (for the event being handled).
  - Ap is the set of *applicable plans* (the relevant plans whose contexts are true).
  - ι, ε, and ρ record a particular intention, event, and applicable plan (respectively) being considered along with the execution of one reasoning cycle.

- The current step within an agent's reasoning cycle is symbolically annotated by $s \in \{\mathsf{ProcMsg}, \mathsf{SelEv}, \mathsf{RelPl}, \mathsf{ApplPl}, \mathsf{SelAppl}, \mathsf{AddIM}, \mathsf{SelInt}, \mathsf{ExecInt}, \mathsf{ClrInt}\}$. These labels stand for, respectively: processing a message from the agent's mail inbox, selecting an event from the set of events, retrieving all relevant plans, checking which of those are applicable, selecting one particular applicable plan (the intended means), adding the new intended means to the set of intentions, selecting an intention, executing the selected intention, and clearing an intention or intended means that may have finished in the previous step.

- The semantics of AgentSpeak makes use of "selection functions" which allow for user-defined components of the agent architecture. We use here only the S_M functions, as originally defined in [151]; the *select message* function is used to select one message from an agent's mail inbox.

In the interests of readability, we adopt the following notational conventions in our semantics rules:

- If C is an AgentSpeak agent circumstance, we write C_E to make reference to the E component of C, and similarly for other components of the multi-agent system and of the configuration of each agent.

- We write AG^{id} to identify the agent represented by that id in the set of agents AG. We use this whenever the component corresponds to a set of tuples $\langle id, \ldots \rangle$. Also, if AG is a set of tuples $\langle id, Conf \rangle$, then we refer to a configuration ($Conf$) of one agent (identified by id) in AG by AG^{id}_{Conf}.

- We write $b[as(asid), d(did), s(id)]$ to identify the origin of a belief related to a dialogue, where $asid$ is the argumentation scheme identifier, did is a dialogue identifier, and id an agent identifier (as refers to argumentation scheme, d refers to $dialogue$ and s refers to $source$). Whenever an agent makes a statement related to a dialogue, the dialogue identifier did is added as an annotation.

- We use two transitions to represent the state change of the multi-agent system, where the transition $\longrightarrow_{AS}$ (transition of the configuration of an individual agent) is part of the transition $\longrightarrow_{DS}$ (the transition of the multi-agent system). So each transition in the agent configuration also causes a transition in the multi-agent system configuration exactly in the component AG^{aid}_{Conf}, where aid refers to the identifier of the agent which went through the transition.

Also, we use aid to refer to the agent that is executing an internal action of interest or receiving a message. Finally, we make use of a function called CTJ (where CTJ stands for "care to justify") that returns `TRUE` if the agent wishes to justify its previous assertion (this depends on the agent's reasoning and makes reference to agents' autonomy).

5.1.3 Semantic Rules for Sending the New Performatives

In this section, we give semantics for sending the new performatives which allow argumentation-based dialogues, showing how it affects the state of the agent and the state of the dialogue.

$$\frac{\begin{array}{c} T_\iota = i[head \leftarrow \texttt{.send}(did, \texttt{assert}, p);h] \\ p \notin CS \qquad \langle aid, CS\rangle \in D_{Ags}^{did} \end{array}}{\begin{array}{ll} (a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \text{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \text{ProcMsg}\rangle \end{array}} \quad \text{(ExecActSndAssert)}$$

where:

$$\begin{array}{llcl} (a) & D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\} \\ & & & \text{with } CS' = CS \cup \{p\} \\ & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \texttt{assert}, p[d(did)]\rangle\} \\ & & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\}) \\ & C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\} \end{array}$$

Internal Action .send **with** assert: The action .send with performative assert updates the CS of the agent that performs the action and sends, to all agents in the dialogue, a message stating that the sender is willing to defend this claim.

The agent can use *assertion attitudes* as defined in [114, 116], but in any case the agent can only assert a formula it did not previously assert; that is, an agent cannot assert again formalæ that are already in its CS. While this semantics for the performative assert formalises the general effects such communication causes on the agent's mental attitudes, later in this document we formalise it according to our framework for argumentation schemes in multi-agent systems.

Another important point to be noticed is that an assertion is always made to a particular dialogue (identified by did) and not to a specific agent; this is because the agents will introduce new claims to be defended to all agents participating in the dialogue and not to an individual agent.

$$\frac{\begin{array}{c} T_\iota = i[head \leftarrow .\mathtt{send}(tid, \mathtt{accept}, p[d(did)]);h] \\ p \notin CS \qquad \langle aid, CS \rangle \in D^{did}_{Ags} \end{array}}{\begin{array}{ll} (a) & \langle AG, D \rangle \longrightarrow_{DS} \langle AG', D' \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt} \rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg} \rangle \end{array}} \text{(ExecActSndAccept)}$$

where:

$$\begin{array}{llcl} (a) & D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS \rangle\}) \cup \{\langle aid, CS' \rangle\} \\ & & & \text{with } CS' = CS \cup \{p\} \\ & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \mathtt{accept}, p[d(did)] \rangle\} \\ & & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\}) \\ & C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\} \end{array}$$

Internal Action .send **with** accept: The action .send with performative accept updates the CS of the agent that performs the action and sends, to all agents in the dialogue, a message stating that the agent accepts the claim made by another agent identified using tid. Note that p (the formula that was accepted) has the annotation $[d(did)]$, this means that p has been previously asserted in that dialogue (identified by did). In other words, an agent can only accept a claim made by another agent in that same dialogue.

$$\frac{\begin{array}{c} T_\iota = i[head \leftarrow .\mathtt{send}(did, \mathtt{retract}, p[d(did)]);h] \\ p \in CS \qquad \langle aid, CS \rangle \in D^{did}_{Ags} \end{array}}{\begin{array}{ll} (a) & \langle AG, D \rangle \longrightarrow_{DS} \langle AG', D' \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt} \rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg} \rangle \end{array}} \text{(ExecActSndRetract)}$$

where:

$$\begin{array}{llcl} (a) & D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS \rangle\}) \cup \{\langle aid, CS' \rangle\} \\ & & & \text{with } CS' = CS \setminus \{p\} \\ & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \mathtt{retract}, p[d(did)] \rangle\} \\ & & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\}) \\ & C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\} \end{array}$$

Internal Action .send **with** retract: The agent performs this internal action to retract a previous claim that the agent itself asserted in that dialogue. The agent's CS is updated with the removal of the given formula. A message is sent to each agent in the dialogue informing the decision of that agent to retract its previous claim.

$$\frac{T_\iota = i[head \leftarrow .\mathtt{send}(id, \mathtt{question}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}} \text{(ExecActSndQuestion)}$$

where:

$$\begin{array}{lll}
(a) \; AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\
(b) \; M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \mathtt{question}, p[d(did)]\rangle\} \\
\quad\; C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}
\end{array}$$

Internal Action `.send` **with** `question`: The action `.send` with performative `question` is used when an agent wants to question another agent in order to know either an argument supporting p or an argument supporting $\neg p$. In both cases, p could have been asserted in that dialogue or not. This message is sent only to the agent which is being questioned. As we will see later, an agent may question an implicit part of an argument using the `question` performative, for example, a critical question related to an argument instantiated from an argumentation scheme known by the agent.

$$\frac{T_\iota = i[head \leftarrow .\mathtt{send}(tid, \mathtt{challenge}, p[d(did)]);h]}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle\end{array}} \text{(ExecActSndChallenge)}$$

where:

$$\begin{array}{lll}
(a) \; D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\} \\
 & & \text{with } CS' = CS \cup \{\neg p\} \\
\quad\; AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\
(b) \; M'_{Out} & = & (M_{Out} \cup \{\langle mid, tid, \mathtt{challenge}, p[d(did)]\rangle\}) \\
 & & \cup \{\langle mid, id, \mathtt{assert}, \neg p[d(did)]\rangle\} \\
 & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid} \cup Ags^{tid}\}) \\
\quad\; C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}
\end{array}$$

Internal Action `.send` **with** `challenge`: The action `.send` with the performative `challenge` is performed when an agent wants to challenge another agent about an assertion it previously made. Differently from the question performative, when an agent makes a challenge move it is willing to defend a claim contrary to the claim of the other agent.

The message with a performative `challenge` is sent only to the agent that made the previous claim. As the agent is willing to defend its claim, messages are sent to all other agents in the dialogue with the respective `assert` messages.

5.1.4 Semantic Rules for Receiving the New Performatives

In this section, we give semantics for receiving the new performatives that allow argumentation-based dialogues, showing how they affect the state of the agent and the state of the dialogue.

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{assert}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Assert)}$$

where:

$$\begin{array}{llcl} (a) & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{assert}, p[d(did)]\rangle\} \\ & ag'_{bs} & = & ag_{bs} + p[d(did), s(sid)] \\ & C'_E & = & C_E \cup \{\langle +p[d(did), s(sid)], \mathrm{T}\rangle\} \end{array}$$

Receiving an `assert` **Message**: The information asserted in the dialogue is added to the belief base of the receiver with an annotation of the dialogue identifier $d(did)$ and the identifier of the agent that asserted the claim as the source of that information $s(sid)$. The agent that received the message can react to this because of the event generated by the belief addition, as usual in AgentSpeak. Whether an agent accepts or not the information received from another agent depends on its *acceptance attitude* as described in [114, 116], which depends on if the agent has or not an acceptable argument to or against that information. Note that, p can be either an argument or an atomic information. When an argument, it can be annotated with the relevant information from our framework for argumentation scheme in multi-agent systems. For example, it could be an argument $\langle S, c\rangle^{\theta}_{sn}$, with S the support of that argument, c its conclusion, sn the argumentation scheme $\langle sn, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$ used to instantiate that argument, and θ the general unifier used.

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{accept}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Accept)}$$

where:

$$\begin{array}{llcl} (a) & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{accept}, p[d(did)]\rangle\} \\ & ag'_{bs} & = & ag_{bs} + p[d(did), s(sid)] \\ & C'_E & = & C_E \cup \{\langle +p[d(did), s(sid)], \mathrm{T}\rangle\} \end{array}$$

Receiving an `accept` **Message**: This message means an agent (identified by sid) accepts a claim previously made, as part of this dialogue, by another agent. The receiver of the message become aware of this acceptance.

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{retract}, p[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Retract)}$$

where:

$$\begin{array}{llcl} (a) & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{retract}, p[d(did)]\rangle\} \\ & ag'_{bs} & = & ag_{bs} - p[d(did), s(sid)] \\ & C'_E & = & C_E \cup \{\langle -p[d(did), s(sid)], \top\rangle\} \end{array}$$

Receiving a `retract` **Message**: This message means an agent, identified by sid, is withdrawing its earlier assertion. The formula is removed from belief base of the receiver of the message, with the appropriate source and dialogue annotation.

$$\frac{\begin{array}{c} S_M(M_{In}) = \langle mid, sid, \texttt{question}, p[d(did)]\rangle \\ \text{CTJ}(p) = \texttt{TRUE} \end{array}}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag, C, M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Question)}$$

where:

$$\begin{array}{llcl} (a) & D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\} \\ & & & \text{with } CS' = CS \cup \{Sp\} \\ & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{question}, p\rangle\} \\ & M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \texttt{justify}, \langle Sp, \psi\rangle[d(did)]\rangle\}, \\ & & & \text{with } Sp \models \psi, \text{ where either } \psi = p \textit{ or } \psi = \neg p \\ & & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\}) \end{array}$$

Receiving a `question` **Message**: If the agent can or wants to reply, in keeping with agent autonomy (this is represented in the semantics through a CTJ function which is meant to be agent specific), then the agent's CS will be updated with the support of the argument for or against p, and the argument will be also sent to all other agents in the dialogue.

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{justify}, \langle Sp, \psi\rangle[d(did)]\rangle}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D, \rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Justify)}$$

where:

$$\begin{array}{llll} (a) & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{justify}, \langle Sp, \psi\rangle[d(did)]\rangle\} \\ & \multicolumn{3}{l}{\text{and for each } p \in Sp :} \\ & ag'_{bs} & = & ag_{bs} + p[d(did), s(sid)] \\ & C'_E & = & C_E \cup \{\langle +p[d(did), s(sid)], \mathsf{T}\rangle\} \end{array}$$

Receiving a `justify` **Message**: This is similar to the assert performative, except for the fact that the content of the message is a set of formulæ that justify the previous claim of the sender of the message (identified by sid). Note that, using our approach for argumentation scheme in multi-agent systems, arguments will be also annotated with the argumentation scheme and the unifier used to instantiate that argument.

$$\frac{\begin{array}{c} S_M(M_{In}) = \langle mid, sid, \texttt{challenge}, p[d(did)]\rangle \\ \text{CTJ}(p) = \texttt{TRUE} \end{array}}{\begin{array}{ll}(a) & \langle AG, D\rangle \longrightarrow_{DS} \langle AG', D'\rangle \\ (b) & \langle ag, C, M, T, \text{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C, M', T, \text{ExecInt}\rangle\end{array}} \quad \text{(Challenge)}$$

where:

$$\begin{array}{llll} (a) & D'^{did}_{Ags} & = & (D^{did}_{Ags} \setminus \{\langle aid, CS\rangle\}) \cup \{\langle aid, CS'\rangle\} \\ & & & \text{with } CS' = CS \cup \{Sp\}; \\ & AG'^{aid}_{Conf} & = & \text{the transition given by (b)} \\ (b) & M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \texttt{challenge}, p[d(did)]\rangle\} \\ & M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \texttt{justify}, Sp[d(did)]\rangle\} \\ & & & \text{for each } Ags_{id} \in (D^{did}_{Ags} \setminus \{Ags^{aid}\}) \\ & & & \text{where } Sp \models p \textit{ and } Sp \in ag_{bs} \\ & ag'_{bs} & = & ag_{bs} + \neg p[d(did), s(sid)] \end{array}$$

Receiving a `challenge` **Message**: If the agent can or wants to reply (we assume the CTJ function determines whether that is the case or not), the agent's CS is updated with the support of the previous claim p and the support is also sent to all other agents in the dialogue. In addition to the question performative, this rule adds that the agent identified by sid (i.e., the sender of the message) is willing to defend the claim contrary to the previous claim that is being challenged.

5.2 Artifacts to Support Argumentation-Based Dialogues

The semantics presented in the previous section can be implemented extending the internal action `.send` (for sending messages) and the `.asl` file (which can be included as part of the agent program) with plans similar to the KQML plans available with Jason. Such AgentSpeak file is used to treat the receiving of messages related to the performatives to which we gave semantics. The corresponding dialogue system can be implemented using a CArtAgO artifact [132] to coordinate agent interactions. Thus, when agents use the speech acts we defined in order to communicate with each other, commitments related to what they say are created in a shared artifact, and all agents participating in such dialogues will have such commitments available to them as observable properties in that artifact. Also, the artifact allows agents to include and remove other participants in dialogues, as well as to execute a broadcast for all participants when needed.

The dialogue artifact provides the following operations:

- `addAgent(Ag)`: This operation adds an agent `Ag` to the dialogue;
- `removeAgent(Ag)`: This operation removes an agent `Ag` from the dialogue;
- `addCS(Content)`: This operation adds the content unified in `Content` to the commitment store of the agent that executes the operation;
- `removeCS(Content)`: This operation removes the content unified in `Content` from the commitment store of the agent executing the operation;
- `allAgents(List)`: This operation returns a list `List` of all agents participating in that dialogue.

Also, we have implemented the commencement rules (rules defining how agents start a dialogue), as predefined agents plans. In order to start a dialogue, an agent only need to create a goal !`opendialogue`($<$ `agents` $>$, $<$ `subject` $>$), specifying with which agents it would like to be part of a dialogue, and the subject of such a dialogue. Thus, this plan follows creating an instance of dialogue, i.e., it adds a new dialogue to the multi-agent system, creating a dialogue artifact with an unique identifier, after it sends messages to all agents with a special performative `opendialogue`, inviting them to participate in the new dialogue, and waits for all agents to reply with either an `acceptdialogue` or a `refusedialogue` message. After all agents have responded, it generates an event called `+continueDialogue(ID)`, with `ID` the dialogue id. An agent can react to `+continueDialogue(ID)` in order to continue the dialogue. Also, we made available a pre-implemented plan !`closedialogue`($<$ `ID` $>$). This plan sends, to all agents in the dialogue to be closed, a message warning them that the dialogue, identified by `ID`, is being closed (using the performative `closedialogue`). The semantic meaning for receiving such messages are as follows:

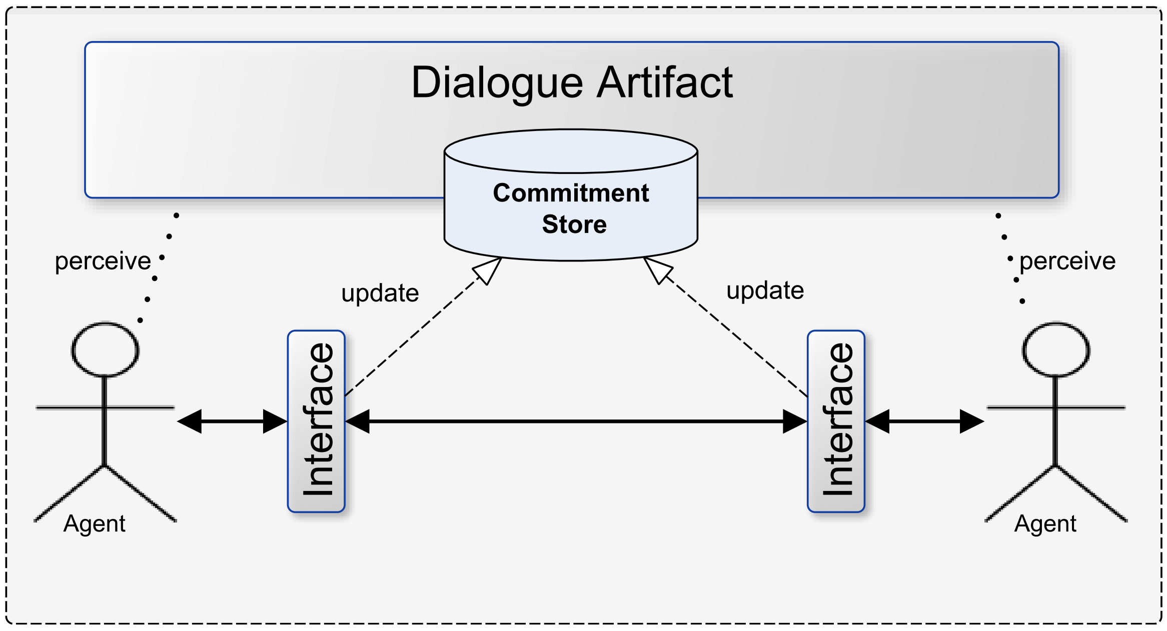


Figure 5.1 – Dialogue Artifact for Argumentation-Based Dialogues.

- `opendialogue`: adds a belief about the dialogue to be started with the source of the agent that proposed the opening of the dialogue (i.e., the sender of the message); The agents that receive the message should react to the event generated, i.e., react to `+dialogue(ID)`, deciding whether to accept or not to participate in the dialogue, and presumably responding with the appropriate message (`acceptdialogue` or `refusedialogue`).

- `closedialogue`: removes the belief about the ongoing dialogue. The dialogue can be closed only by the same agent (identified by *id*) which previously opened the dialogue.

- `acceptdialogue`: the sender of the message is removed from the set of agents that are expected to respond. If all agents respond, the intention can be resumed, otherwise the intention remains suspended.

- `refusedialogue`: the sender of the message is removed from the set of agents that are expected to respond, as well as from the set of agents participating in the dialogue. If all agents have replied, the intention can be resumed.

Besides the performatives agents are able to use in argumentation-based dialogues, agents might follow some rules established by protocols. Protocols are used to restrict the moves agent can do (where moves correspond to the speech-act they are able to use), and the strategy of the agents can be implemented as AgentSpeak plans. The strategy for the agents can be rewritten or replaced easily. As an example of protocol we mention our previous work [110], where we defined a protocol for task reallocation in a collaborative scenario [106, 136].

In order to facilitate the implementation of different protocols and strategies, we have implemented a set of inference rules that programmers can use through queries. Among the more useful queries are:

- `has_argument(Content, Justification)`: using this predicate, it is possible to query if an agent has an argument supporting a particular conclusion.
- `has_argument_against(Content, Justification)`: using this predicate, it is possible to query if an agent has an argument against a particular conclusion.
- `inCS(Content)`: using this predicate, it is possible to query if the agent, which is executing that query, has already included that information, unified in `Content`, in its commitment store.
- `inCS(Content, AgentName)`: using this predicate, it is possible to query if an agent, which its name is unified in `AgentName`, has that information, unified in `Content`, in its commitment store.
- `isSubject(Content)`: using this predicate, it is possible to query the subject of the dialogue.

With this set of pre-implemented queries, pre-implemented plans, and semantics rules, we aim to facilitate the implementation of different protocols and strategies, which will enable different types of argumentation-based dialogues.

5.3 Example

In this section, we describe how our argumentation-based reasoning mechanism presented in the previous chapter can be used in agent communication. Aimed for decentralised systems, as multi-agent systems are supposed to be, we consider that agents use the argumentation-based reasoning mechanism to decide the acceptability of the information received in argumentation-based dialogues. Therefore, when an agent receives a claim or an argument, each information needs to be processed in order to check if that information is acceptable or not given the current state of the agent's own belief base.

For example, we can consider the agent *acceptance attitudes* introduced in [116, 114] adapted for our argumentation framework as follows: (i) a *credulous* agent accepts any claim c if it is unable to construct an acceptable argument against c, i.e., it does not has an acceptable argument $\langle S, \overline{c} \rangle$; and (ii) a *sceptical* agent accepts any claim c if it has an acceptable argument supporting c, i.e., it has an acceptable argument $\langle S, c \rangle$.

In Jason, the handling of received messages is done by special plans triggered by the architecture layer as[5] `+!msg_received(Sender,Performative,Content)`. These plans can be reimplemented to suit particular multi-agent system domains. The *acceptance attitudes* described above, can be implemented as follows:

[5]Such a triggering event is customisable in the Jason configuration files.

```
+!msg_received(Sender,assert,Content):
    not(has_argument_againt(Content,Arg)) &
    acceptance_attitude(credulous)
<- +Content[source(Sender)]; .send(Sender,accept,Content).

+!msg_received(Sender,assert,Content):
    has_argument(Content,Arg) &
    acceptance_attitude(sceptical)
<-  +Content[source(Sender)]; .send(Sender,accept,Arg).
```

When the agent wants to assert some claim in a dialogue, following the same approach suggested in [116, 114], the agent could have two *assertion attitudes* adapted to our framework: (i) a *confident* agent asserts a claim c if it does not have an acceptable argument against c, i.e., it does not have an acceptable argument $\langle S, \overline{c} \rangle$; and (ii) a *thoughtful* agent asserts a claim c only if it has an acceptable argument supporting c, i.e., it has an acceptable argument $\langle S, c \rangle$. These *assertion attitudes* can be implemented as follows:

```
+!assert(Sender,Content):
    not(has_argument_against(Content,Arg)) &
    assertion_attitude(confident)
<- .send(Sender,assert,Content).

+!assert(Sender,Content):
    has_argument(Content,Arg) &
    assertion_attitude(thoughtful)
<- .send(Sender,assert,Content).
```

The plans above are used to exemplify the use of the argumentation-based reasoning mechanism of our framework in argumentation-based communication. While we demonstrate the implementation of some agent *attitudes* from [116, 114], more sophisticated plans can be implemented in order to enable interesting argumentation-based dialogues, which could include storing the moves and assertions done by agents in dialogue, checking the commitments already done in the dialogue using the predicate `inCS(Content)`, for example.

For example, we have implemented a protocol for task reallocation, presented in [110]. The protocol was defined using the well-known speech acts from the literature in argumentation [8, 7, 114, 116, 81] and the CArtAgO artifact we developed, also presented in [97]. Also, the operational semantics of the speech acts for agents based on the BDI mental attitudes presented in this section was published [111, 108]. Such a protocol, artifacts, and operational semantics definition were developed considering argumentation-based dialogues between agents without considering argumentation schemes, therefore, we do not describe such work here in depth.

5.4 Argumentation-Based Dialogues with Argumentation Schemes

In an argumentation-based dialogue using argumentation schemes, agents can consider the critical questions (explicitly available to them towards the argumentation schemes) as extra moves allowed in such dialogues. For example, for the argumentation scheme *Argument from role to know in multi-agent systems* introduced above, the correspondent critical questions are internally represented by agents as special predicates `cq(cq`$_{id}$`,Content)[as(as`$_{id}$`)]`:

```
cq(cq1,role_to_know(Role,Conclusion))[as(role_to_know)]
cq(cq2,honest(Agent))[as(role_to_know)]
cq(cq3,asserts(Agent,Conclusion))[as(role_to_know)]
cq(cq4,role(Agent,Role))[as(role_to_know)]
```

Therefore, when an agent receives a justify message with the content labelled with an argumentation scheme, it is able to check all the critical questions available to that scheme and to use the appropriated ones to question the other agents. Consider the example below[6]:

```
+!msg_received(Sender,assert,⟨S,c⟩^θ[as(AS)]) :
    not(has_argument_against(c,Arg)) &
    cq(Cq,Content)[as(AS)] & not(Content)
<-  .send(Sender,question,Content).
```

In this example, the agent receives an argument annotated with `as(AS)` meaning that the conclusion of this argument `c` is inferred from a particular argumentation scheme `AS`. If the agent does not have an acceptable argument against `c` (of course this could depend on the agent profile), it can check the critical questions for which it has no answers — `cq(Cq,Content)[as(AS)] & not(Content)` — then question the `Sender` about such critical questions in the argumentation scheme used to infer `c`.

5.5 Evaluation

In order to evaluate our framework for argumentation-based dialogues using argumentation scheme in multi-agent systems, we have defined two protocols using argumentation schemes, and we have implemented these protocols using our framework. We show that our framework can be used to implement different protocols, in which the modules and artifacts we implemented, described in Section 5.2, support this development.

In Section 5.5.1, we start describing a framework we use to model argumentation-based dialogues based on dialogue games. After, in Section 5.5.2, we describe a general persuasion protocol we extended from our previous work [110] in order to consider argumentation schemes. Finally,

[6]The example was simplified for showing the role of critical questions in dialogues.

in Section 5.5.3, we describe a protocol for data access control, which is based on the argumentation schemes for data access control we introduced in Section 4.5.2. For both protocols, we will show different dialogues outputs from our implementation, given different agent configurations.

5.5.1 A framework for Argumentation-Based Dialogues

In this section, we describe a framework to specify dialogue games. In our previous work, published in [110], we have used this framework to model argumentation-based dialogues. Here, we use the same framework to specify two different argumentation-based protocols based on our approach for argumentation schemes in multi-agent systems. The framework for dialogue games is based on the work of McBurney *et al.* [80, 79], in which the elements that correspond to the dialogue game specification are:

- **Commencement Rule**: it describes the condition for an agent to start a dialogue.
- **Locutions**: it describes the set of locutions, in our case the locutions from Section 5.1.1, an agent is allowed to use, following a particular protocol.
- **Combination Rules**: it describes the possible combination of locutions. Normally, it depends on the strategy of the agent (e.g., corresponding the agent attitudes to assert and accept claims in the dialogue [116, 114]).
- **Commitments**: it describes the update of commitments of the participants following the semantics of the speech acts used, in our case the updates described in Section 5.1, where each speech-act/performative introduces commitments of participants.
- **Termination Rules**: it describes when a dialogue ends.

In particular, *Dialogue rules* will govern the interactions between the agents, where each agent moves by performing the allowed utterances. These rules (which correspond to a dialogue game [80]) are expressed as if-then rules, which are then easy to implement. The dialogue rules specify the moves each player can make, and so specify the *protocol* under which the dialogue takes place [7].

Definition 20 (Dialogue Game Protocol [110]). *A dialogue game protocol is formally represented as a tuple* $\langle \mathtt{MO}, \mathtt{DI} \rangle$*, where* $\mathtt{MO}$ *is a finite set of allowed moves, and* $\mathtt{DI}$ *a set of dialogue rules.*

Definition 21 (Dialogue Move [110]). *We denote a move in* $\mathtt{MO}$ *as* $\mathtt{M}^{\mathtt{i}}(\alpha, \beta, \mathtt{content}, \mathtt{t})$*, where* $\mathtt{i}$ *is the type of move made by agent* α *and addressed to agent* β *at time* $\mathtt{t}$ *regarding content* `content`*. We consider the following set of types of moves, denoted by* $\mathtt{P}$ *(q.v. Section 5.1.1):* `assert`, `accept`, `question`, *and* `justify`*. The content of a move (*`content`*) can be an argument (a set of predicates and rules) or a single predicate.*

The dialogue rules in $\mathtt{DI}$ indicate the possible moves that an agent can make following a previous move of the other agent. The formalisation we give here follows the work of Bentahar *et al.* [16]. To define the dialogue rules, we use a set of condition (denoted by $\mathtt{C}$) which reflect the agents' strategies. Formally, we have:

Definition 22 (Dialogue Rules'[110]). *Dialogue rules can assume one of two forms:*

1) First, we have dialogue rules that specify which moves are allowed given the previous move and conditions (corresponding to the combination rules of the dialogue game).

$$\bigwedge_{\substack{0<\mathtt{k}\leq \mathtt{n_i},\\ \mathtt{i,j}\in\mathtt{P}}} (\mathtt{M^i}(\alpha, \beta, \mathtt{content}, \mathtt{t}) \wedge \mathtt{C_k} \Rightarrow \mathtt{M^j_k}(\beta, \alpha, \mathtt{content_k}, \mathtt{t'})$$

where $\mathtt{P}$ *is the set of move types,* $\mathtt{M^i}$ *and* $\mathtt{M^j}$ *are in* $\mathtt{MO}$, $\mathtt{t} < \mathtt{t'}$ *and* $\mathtt{n_i}$ *is the number of allowed communicative acts that* β *could perform after receiving a move of type* $\mathtt{i}$ *from* α.

2) Second, we have the initial conditions (corresponding to the commencement rules of the dialogue game), which do not require that any move was previously executed.

$$\bigwedge_{\substack{0<\mathtt{k}\leq \mathtt{n},\\ \mathtt{j}\in\mathtt{P}}} (\mathtt{C_k} \Rightarrow \mathtt{M^j_k}(\alpha, \beta, \mathtt{content_k}, \mathtt{t_0})$$

where $\mathtt{t_0}$ *is the initial time and* $\mathtt{n}$ *is the number of allowed moves that* α *could make initially.*

Using this framework for dialogue games, we will define two different protocols, considering argumentation scheme. After, we show how our framework for argumentation schemes in multi-agent systems can be used to specify dialogue rules, and how we implement those protocols (and respective dialogue rules) in Jason's agent.

In order to define the dialogue rules we use the following notation:

- We write Δ_{ag} to represent the agent ag's knowledge base. Δ_{ag} includes ag's private knowledge, argumentation scheme and organisational knowledge, i.e., $\Delta_{\mathtt{AS}} \cup \Delta_{\mathtt{Org}} \subset \Delta_{\mathrm{ag}}$.

- We write $\mathtt{CS}_{\mathrm{ag}}$ to represent the agent ag's commitment store, which is updated according the semantics rules described in Section 5.1.

- We write $\Delta_{\mathrm{ag}} \models \langle S, c\rangle^{\theta}_{sn}$ to represent that an agent ag is able to instantiate an acceptable argument supporting c (according Definition 19) from the information available in its knowledge base Δ_{ag}, which includes the argumentation scheme sn used to instantiate that argument.

- We write $(\Delta_{ag_i} \cup \mathtt{CS}_{ag_j}) \models \langle S, c\rangle^{\theta}_{sn}$ to represent that an agent ag_i is able to construct an acceptable argument supporting c from its knowledge base and ag_j's commitment store.

5.5.2 A Protocol for Persuasion

A persuasion dialogue is used when an initiator agent believes something that it wants to convince another agent to believe [139]. In this type of dialogue, an agent starts the dialogue with an `assert` move, asserting the information it wants the other agent to believe (the subject of the dialogue) [110]. After, agents are able to present their arguments supporting or attacking the subject of the dialogue. In the end, either the other agent accepts the subject of the dialogue, when the arguments from the initiator agent convince it, or the initiator agent accepts that it cannot convince the other agent to accept the subject of the dialogue, closing the dialogue.

In the previous sections, we have used examples of arguments typically used in persuasion dialogues, for example, the ones instantiated from the argumentation scheme ***argument from role to know***. For example, in a hospital, nurses usually use arguments based on the doctors' assertions in order to convince patients to accept treatment, a diagnostic, etc.

In order to evaluate our framework, we have implemented the following ***persuasion*** protocol, which is an extended version of [110], making reference to components of our framework for argumentation schemes in multi-agent systems:

1. an agent ag_i starts a dialogue with another agent ag_j, executing an `assert` move `assert`(ag_i, ag_j, p), trying to convince ag_j to believe in p. p becomes the subject of the dialogue (the protocol goes to (2)).

2. an agent ag_j receives an `assert` move `assert`(ag_i, ag_j, p), and it checks if it has an argument against p. When it has no argument against p, it accepts p executing an `accept` move `accept`(ag_j, ag_i, p) (the protocol goes to (5)). Otherwise, when ag_j is able to construct an argument against p, it executes a `question` move `question`(ag_j, ag_i, p), asking for an argument supporting that assertion (the protocol goes to (3)).

3. an agent ag_i receives a `question` move `question`(ag_j, ag_i, p), and it responds with a `justify` move `justify`$(ag_i, ag_j, \langle S, p\rangle^{\theta}_{sn})$, introducing an argument which supports the questioned proposition (the protocol goes to (4)).

4. an agent ag_j receives a `justify` move `justify`$(ag_i, ag_j, \langle S, p\rangle^{\theta}_{sn})$, and it checks if the new information received from ag_i convinces it to accept the subject of the dialogue, i.e., if it has no argument against the subject of the dialogue, considering the information received from ag_i. In case ag_j has no argument against the subject of the dialogue, it accepts the subject of the dialogue executing an `accept` move `accept`(ag_j, ag_i, p) (the protocol goes to (5)). Otherwise, when ag_j is able to construct an argument against the subject of the dialogue, either (i) it executes an `assert` move `assert`$(ag_j, ag_i, \neg p)$, committing itself to support that $\neg p$ is the case[7] (the protocol goes to (2)); or (ii) it checks if it is able to answer negatively any

[7]Note that ag_j is able to execute this move only when it did not assert $\neg p$ previously, i.e., $\neg p \notin CS_{ag_j}$.

critical question related to the argumentation scheme sn that ag_i has used to instantiate its argument. In the case it is able to answer negatively a critical question from sn, it executes an `assert` move $\texttt{assert}(ag_j, ag_i, \neg cq)$, with cq the critical question (the protocol goes to (2)).

5. an agent ag_j receives an `accept` move $\texttt{accept}(ag_i, ag_j, p)$, and in the case the accepted proposition is the subject of the dialogue, the dialogue ends with both agents believing in the subject of the dialogue. In the case the accepted proposition is a critical question, the agent which accepts the critical question returns to step (4) in the protocol, accepting the subject of the dialogue, questioning another critical question, or committing itself to support that the subject of the dialogue is not the case.

Dialogue Rules:

- **Initial Rule**: The first move (commencement rule) introduces the subject of the dialogue (where `subject(p)` means that the predicate `p` is the subject of the dialogue). Each dialogue has only one `subject`. The agent that introduces the subject of the dialogue is called ***proponent***, and the other agent participating in the dialogue is called ***opponent*** [41] (we use `Pr` and `Op`, respectively, to refer to them).

  $$\texttt{C}_{in1} \Rightarrow \texttt{assert}(ag_i, ag_j, \texttt{p})$$

  where:

  $\texttt{C}_{in1} = \exists \langle S, p \rangle_{sn}^{\theta} : \Delta_{ag_i} \models \langle S, p \rangle_{sn}^{\theta}$

  The dialogue starts when an agent wants to argue about a given subject. The initial rule restricts that an agent needs to have an argument that supports its claim in order to start an argumentation-based dialogue (as the agent will be committed to defending the initial assertion). Considering our framework for argumentation scheme in multi-agent systems, it requires the agent being able to instantiate an acceptable instance of argument from an argumentation scheme in $\Delta_{\texttt{AS}}$.

- **Assert Rules**: We have two dialogue rules that restrict the possible next move for agents to respond to an `assert` move:

  $$\texttt{assert}(ag_i, ag_j, \texttt{p}) \land \texttt{C}_{as1} \Rightarrow \texttt{accept}(ag_j, ag_i, \texttt{p})$$

  $$\texttt{assert}(ag_i, ag_j, \texttt{p}) \land \texttt{C}_{as2} \Rightarrow \texttt{question}(ag_j, ag_i, \texttt{p})$$

  where:

  $\texttt{C}_{as1} = \nexists \langle S, \overline{p} \rangle_{sn}^{\theta} : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \langle S, \overline{p} \rangle_{sn}^{\theta}$

  $\texttt{C}_{as2} = \exists \langle S, \overline{p} \rangle_{sn}^{\theta} : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \langle S, \overline{p} \rangle_{sn}^{\theta}$

The options of the agent are: (i) to accept the previous claim, where condition $\mathtt{C}_{as1}$ means that the agent will accept a claim if it has no argument against it; and (ii) when the agent has an argument against the previous assertion, $\mathtt{C}_{as2}$, the agent will question the other agent to provide the support of its previous claim.

- **Question Rule**: The dialogue rule that restricts the moves after an agent receives a `question` message is:

$$\mathtt{question}(ag_i, ag_j, \mathrm{p}) \wedge \mathtt{C}_{qs1} \Rightarrow \mathtt{justify}(ag_j, ag_i, \langle S, p\rangle^{\theta}_{sn})$$

Where:

$\mathtt{C}_{qs1} = \exists\langle S, p\rangle^{\theta}_{sn} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S, p\rangle^{\theta}_{sn}$

Considering that the agent has asserted a predicate p previously (which allowed the `question` move), the agent is committed to defending its claim in the dialogue, so it will provide the support to this claim.

- **Justify Rules**: We have four dialogue rules that restrict the possible next move for agents to respond to a `justify` move:

$$\mathtt{justify}(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}) \wedge \mathtt{C}_{js1} \Rightarrow \mathtt{accept}(ag_j, ag_i, \mathrm{p})$$

$$\mathtt{justify}(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}) \wedge \mathtt{C}_{js2} \Rightarrow \mathtt{assert}(ag_j, ag_i, \overline{p})$$

$$\mathtt{justify}(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}) \wedge \mathtt{C}_{js3} \Rightarrow \mathtt{assert}(ag_j, ag_i, \overline{cq})$$

$$\mathtt{justify}(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}) \wedge \mathtt{C}_{js4} \Rightarrow \mathtt{closedialogue}(ag_j, ag_i)$$

$$\mathtt{justify}(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}) \wedge \mathtt{C}_{js5} \Rightarrow \mathtt{justify}(ag_j, ag_i, \mathrm{S}')$$

where:

$\mathtt{C}_{js1} = \nexists\langle S', \overline{p}\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', \overline{p}\rangle^{\theta}_{sn_j} \wedge\ \mathtt{Op}(ag_j) \wedge \mathtt{subject}(p)$

$\mathtt{C}_{js2} = \exists\langle S', \overline{p}\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', \overline{p}\rangle^{\theta}_{sn_j} \wedge \overline{p} \notin \mathtt{CS}_{ag_j} \wedge\ \mathtt{Op}(ag_j) \wedge \mathtt{subject}(p)$

$\mathtt{C}_{js3} = \exists\overline{cq}\theta : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \overline{cq}\theta \wedge \overline{cq}\theta \notin \mathtt{CS}_{ag_j} \wedge cq\theta \in \mathcal{CQ} \wedge \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{\mathtt{AS}}$

$\mathtt{C}_{js4} = \nexists\langle S', p\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', p\rangle^{\theta}_{sn_j} \wedge \langle S', p\rangle^{\theta}_{sn_j} \notin \mathtt{CS}_{ag_j} \wedge\ \mathtt{Pr}(ag_j)\ \wedge \mathtt{subject}(p)$

$\mathtt{C}_{js5} = \exists\langle S', p\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', p\rangle^{\theta}_{sn_j} \wedge \langle S', p\rangle^{\theta}_{sn_j} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Pr}(ag_j) \wedge \mathtt{subject}(p)$

The agent will accept the subject of the dialogue, $\mathtt{C}_{js1}$, if the justification received from the proponent has changed the agent's conclusion, otherwise either the agent will assert that it is committed to supporting a claim against the subject of the dialogue when it has an argument against it, $\mathtt{C}_{js2}$, or it will assert that some critical question related to that argument are not positively answered, $\mathtt{C}_{js3}$. In the case in which the agent that receives the justify move from the opponent cannot itself reach the same conclusion, given the new information received (i.e., the agent does not have an acceptable argument for the subject of the dialogue anymore), the agent closes the dialogue, $\texttt{C}_{js4}$. In the final case, the agent sends a new argument[8] to support the subject of the dialogue, $\texttt{C}_{js5}$.

- **Accept Rule**: The dialogue rule that restricts the moves when an agent receives an `accept` message is:

  $$\texttt{accept}(ag_i, ag_j, \texttt{p}) \wedge \texttt{C}_{ac1} \Rightarrow \texttt{closedialogue}(ag_j, ag_i)$$

  where:

  $\texttt{C}_{ac1} = \texttt{subject}(\texttt{p}) \wedge \texttt{Pr}(ag_j)$

  When the agent receives an accept move it will close the dialogue. Only the proponent will receive an accept move when the opponent accepts the subject of the dialogue.

Implementation in Jason Agents

The dialogue rules can be easily implemented in Jason platform. When an agent receives a message, an event of the type `+!msg_received(Sender, Performative, Content)` is generated to the receiver agent, and the agent can treat that event generating new goals. Thus, the dialogue rules presented in this section can be implemented towards agents plans that aim to respond to received messages.

The number of plans to treat receiving a particular message will be equal to the number of dialogue rules that restrict the next possible move the agent can respond. For example, the **assert rules** presented in this section can be implemented using the following agent's plans:

```
+!respondAssert(Sender,Content):
            not(has_argument_against(Content,Arg))
        <- !accept(Sender,Content).

+!respondAssert(Sender,Content): has_argument_against(Content,Arg)
        <- !question(Sender,Content).
```

Similarly, other condition used in dialogue rules can be easily implemented using our framework through the modules and artifacts we implemented, described in Section 5.2.

Experiments

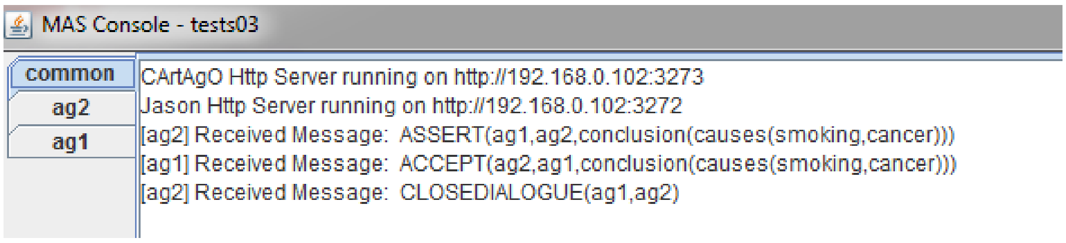
We ran some experiments to show different outputs from the protocol for persuasion implemented in our framework. In our experiments, two agents, named $ag1$ and $ag2$, use

[8]The argument is new because, as defined in the protocol, the agent cannot repeat a move with the same content.

the argumentation scheme *role to know* in order to argue about whether *smoking causes cancer* or not, based on doctors' assertions. Thus, the agent $ag1$ starts a dialogue asserting that $\mathtt{causes(smoking, cancer)}$, which becomes the subject of the dialogue. The ag_1's knowledge is represented in Δ_{ag1}

$$\Delta_{ag1} = \left\{ \begin{array}{l} \mathtt{reliable(john)} \\ \neg\mathtt{reliable(pietro)} \\ \mathtt{asserts(john, causes(smoking, cancer))} \\ \mathtt{role(john, doctor)} \\ \mathtt{role\_to\_know(doctor, cancer)} \\ \mathtt{about(causes(smoking, cancer), cancer).} \end{array} \right\}$$

In the case that the agent $ag2$ has no argument against the subject of the dialogue, i.e., $\mathtt{causes(smoking, cancer)}$, it accepts it, as show the output from our implementation in Figure 5.2.



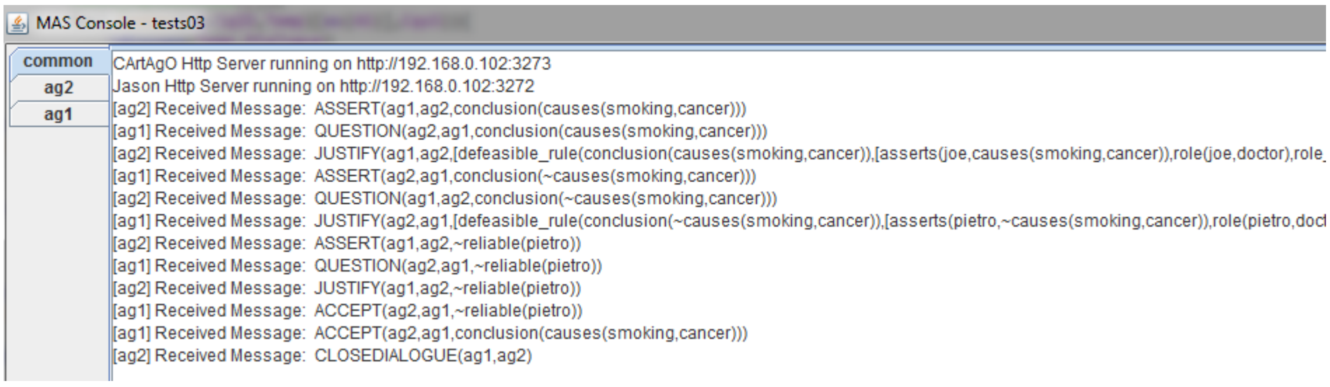
```
CArtAgO Http Server running on http://192.168.0.102:3273
Jason Http Server running on http://192.168.0.102:3272
[ag2] Received Message:  ASSERT(ag1,ag2,conclusion(causes(smoking,cancer)))
[ag1] Received Message:  ACCEPT(ag2,ag1,conclusion(causes(smoking,cancer)))
[ag2] Received Message:  CLOSEDIALOGUE(ag1,ag2)
```

Figure 5.2 – First Example

In the case that the agent $ag2$ has an argument against the subject of the dialogue, we have a different output.

$$\Delta_{ag2} = \left\{ \begin{array}{l} \mathtt{asserts(pietro, \neg causes(smoking, cancer))} \\ \mathtt{role(pietro, doctor)} \\ \mathtt{role\_to\_know(doctor, cancer)} \\ \mathtt{about(causes(smoking, cancer), cancer).} \end{array} \right\}$$

Considering the $ag2$'s knowledge represented in Δ_{ag2}, in which ag_2 assumes that *Pietro* is a reliable doctor by assumption, it initially has an argument against the subject of the dialogue. But, after receiving the information that Pietro is not a reliable doctor from ag_1, it accepts the subject of the dialogue. The output of this dialogue is shown in Figure 5.3.



```
CArtAgO Http Server running on http://192.168.0.102:3273
Jason Http Server running on http://192.168.0.102:3272
[ag2] Received Message:  ASSERT(ag1,ag2,conclusion(causes(smoking,cancer)))
[ag1] Received Message:  QUESTION(ag2,ag1,conclusion(causes(smoking,cancer)))
[ag2] Received Message:  JUSTIFY(ag1,ag2,[defeasible_rule(conclusion(causes(smoking,cancer)),[asserts(joe,causes(smoking,cancer)),role(joe,doctor),role_
[ag1] Received Message:  ASSERT(ag2,ag1,conclusion(~causes(smoking,cancer)))
[ag2] Received Message:  QUESTION(ag1,ag2,conclusion(~causes(smoking,cancer)))
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(conclusion(~causes(smoking,cancer)),[asserts(pietro,~causes(smoking,cancer)),role(pietro,doct
[ag2] Received Message:  ASSERT(ag1,ag2,~reliable(pietro))
[ag1] Received Message:  QUESTION(ag2,ag1,~reliable(pietro))
[ag2] Received Message:  JUSTIFY(ag1,ag2,~reliable(pietro))
[ag1] Received Message:  ACCEPT(ag2,ag1,~reliable(pietro))
[ag1] Received Message:  ACCEPT(ag2,ag1,conclusion(causes(smoking,cancer)))
[ag2] Received Message:  CLOSEDIALOGUE(ag1,ag2)
```

Figure 5.3 – Second Example

5.5.3 A Protocol for Data Access Control

In this section, we propose a protocol for data access control in multi-agent systems, based on the argumentation schemes for data access control described in Section 4.5.2. In this section, we start describing the dialogue game specification for this protocol. After, we describe its implementation in our framework.

Different from the previous protocol, this protocol was specified based on argumentation schemes used in a particular application domain, i.e., argumentation schemes for data access control. Consequently, the protocol for data access control is more simple than a protocol for persuasion. Basically, a requester agent will request access to a particular piece of information, and an interface agent will provide or not the information according to the data access control rules of that system, and who is the requester. After, in the case that the access to the information was denied, the requester agent is able to provide more information about itself, or about any emergency situation that could change that decision. The protocol is as follows:

1. an agent ag_i (the requester) opens a dialogue with another agent ag_j (the interface agent) executing a `question` move $\mathtt{question}(ag_i, ag_j, \mathtt{access}(\mathtt{ag_i}, \phi))$, requesting access to information ϕ (the protocol goes to (2)).

2. an agent ag_j (the interface agent) checks if it has an acceptable argument supporting $\mathtt{access}(ag_i, \phi)$. In the case it has an acceptable argument, it grants access to that information, executing an `accept` move $\mathtt{accept}(ag_j, ag_i, \mathtt{access}(\mathtt{ag_i}, \phi\theta))$ (the protocol goes to (3)). In the case, ag_j is not able to construct an acceptable argument to $\mathtt{access}(a_1, \phi)$, it executes a `justify` move $\mathtt{justify}(ag_j, ag_i, \langle S, \neg\mathtt{access}(\mathtt{ag_i}, \phi)\rangle^{\theta}_{[\mathtt{as(as4dac)}]})$, with $S = \{\mathtt{inf\_category}(\phi, c_1), \mathtt{ac\_category}(ag_i, r_1), \neg\mathtt{access}(r_1, c_1), [\mathtt{inf\_category}(I, C), \mathtt{ac\_category}(A, R), \neg\mathtt{access}(R, C) \Rightarrow \neg\mathtt{access}(A, I)]\}$, justifying that the access-category of ag_i has no access to the information-category of ϕ (the protocol goes to (4)).

3. an agent ag_i closes the dialogue.

4. an agent ag_i decides to: (i) provide more information, either about itself in order to be categorised into a different access-category, or about an emergency situation in order to make ag_j to consider the emergency access control rules — for both cases ag_i executes an `assert` move $\texttt{assert}(ag_i, ag_j, \varphi\theta)$, with $\varphi\theta$ the information ag_i wants to be considered by ag_j (the protocol goes to (5)); (ii) accept that it has no access to information ϕ (the protocol goes to (3)).

5. an agent ag_j adds the information received to its knowledge base (the protocol goes to (2)).

Dialogue Rules

- **Initial Rule**: The first move (commencement rule) introduces a request from an agent ag_i (the requester), requesting access to a particular piece of information ϕ, to another agent ag_j (the interface agent) which will decide to provide or not the requested information to ag_i.

  $$\mathtt{C}_{in1} \Rightarrow \texttt{question}(ag_i, ag_j, \texttt{access}(\texttt{ag}_\texttt{i}, \phi))$$

  where: $\mathtt{C}_{in1} = \texttt{true}$

  Note that there is no condition to execute this move at the level of dialogue rules specification. We could argue that an agent might have its motivation to obtain that information, but it is part of the agent strategy and attitude towards participating in such dialogue.

- **Question Rule**: We have two dialogue rules that restrict the moves after an agent receives a `question` message:

  $$\texttt{question}(ag_i, ag_j, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)) \wedge \mathtt{C}_{qs1} \Rightarrow \texttt{accept}(ag_j, ag_i, \texttt{access}(\texttt{ag}_\texttt{i}, \phi\theta))$$

  $$\texttt{question}(ag_i, ag_j, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)) \wedge \mathtt{C}_{qs2} \Rightarrow \texttt{justify}(ag_j, ag_i, \langle \mathtt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]})$$

  where:

  $\mathtt{C}_{qs1} = \exists\langle \mathtt{S}, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(as4dac)}]} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle \mathtt{S}, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(as4dac)}]}$

  $\mathtt{C}_{qs2} = \exists\langle \mathtt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle \mathtt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]}$

  Basically, the interface agent provides the information when it has an acceptable argument concluding that the request agent has access to the requested information, $\mathtt{C}_{qs1}$. This acceptable argument is an instance from the argumentation scheme for data access control, `as4dac`. Otherwise, it will have an acceptable argument concluding that the requester agent has no access to that information, thus it justifies to the requester why it has not access to the requested information, $\mathtt{C}_{qs2}$.

- **Justify Rules**: We have three dialogue rules that restrict the possible next move for agents to respond to a justify move:

$$\texttt{justify}(ag_i, ag_j, \langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]}) \wedge \texttt{C}_{js1} \Rightarrow \texttt{assert}(ag_j, ag_i, \texttt{role}(\texttt{ag}_\texttt{j}, \varphi))$$

$$\texttt{justify}(ag_i, ag_j, \langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]}) \wedge \texttt{C}_{js2} \Rightarrow \texttt{assert}(ag_j, ag_i, \texttt{emrg}(\varphi))$$

$$\texttt{justify}(ag_i, ag_j, \langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]}) \wedge \texttt{C}_{js3} \Rightarrow \texttt{closedialogue}(ag_j, ag_i)$$

where:

$\texttt{C}_{js1} = \exists\ \texttt{role}(\texttt{ag}_\texttt{j}, \varphi) : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \texttt{role}(\texttt{ag}_\texttt{j}, \varphi) \wedge \texttt{role}(\texttt{ag}_\texttt{j}, \varphi') \in S \wedge \texttt{role}(\texttt{ag}_\texttt{j}, \varphi') \neq \texttt{role}(\texttt{ag}_\texttt{j}, \varphi) \wedge \texttt{role}(\texttt{ag}_\texttt{j}, \varphi) \notin \texttt{CS}_{ag_j}$

$\texttt{C}_{js2} = \exists\ \texttt{emrg}(\varphi) : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \texttt{emrg}(\varphi) \wedge \texttt{emrg}(\varphi) \notin \texttt{CS}_{ag_j}$

$\texttt{C}_{js3} = \texttt{true}$

The options of the requester agent are: (i) to assert the correct role it plays in the multi-agent systems it belongs, when the interface agent has used a different role, $\texttt{C}_{js1}$; (ii) to assert an emergency situation in which an emergency policy could apply for that particular case, $\texttt{C}_{js2}$; (iii) to close the dialogue, accepting that it has no access to that information.

- **Assert Rules**: We have two dialogues rules that restrict the possible next move for agents to respond to an assert move:

$$\texttt{assert}(ag_i, ag_j, \texttt{role}(\texttt{ag}_\texttt{i}, \varphi)/\texttt{emrg}(\varphi)) \wedge \texttt{C}_{as1} \Rightarrow \texttt{accept}(ag_j, ag_i, \texttt{access}(\texttt{ag}_\texttt{i}, \phi\theta))$$

$$\texttt{assert}(ag_i, ag_j, \texttt{role}(\texttt{ag}_\texttt{i}, \varphi)/\texttt{emrg}(\varphi)) \wedge \texttt{C}_{as2} \Rightarrow \texttt{justify}(ag_j, ag_i, \langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]})$$

where:

$\texttt{C}_{as1} = \exists\langle \texttt{S}, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(as4dac)}]} : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \langle \texttt{S}, \texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(as4dac)}]}$

$\texttt{C}_{as2} = \exists\langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]} : (\Delta_{ag_j} \cup \texttt{CS}_{ag_i}) \models \langle \texttt{S}, \neg\texttt{access}(\texttt{ag}_\texttt{i}, \phi)\rangle^{\theta}_{[\texttt{as(comp\_as4dac)}]}$

The conditions $\texttt{C}_{as1}$ and $\texttt{C}_{as2}$ are similar to the question move. When the interface agent has an argument concluding that the requester agent has access to that information, $\texttt{C}_{as1}$, it will provide the information. When it has an argument concluding that the requester agent has no access to that information, $\texttt{C}_{as2}$, it will justify why the requester agent has no access to that information.

- **Accept Rules**: We have one dialogue rule that restrict the next move for agents to respond to an accept move:

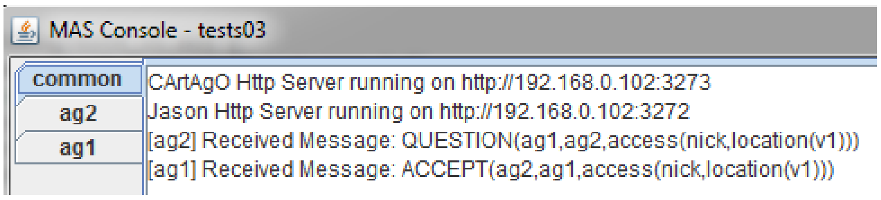
$$\texttt{accept}(ag_i, ag_j, \texttt{access}(\texttt{ag}_\texttt{j}, \phi\theta)) \Rightarrow \texttt{closedialogue}(ag_j, ag_i)$$

There is no condition for an agent close the dialogue after receiving the requested information.

Experiments

We evaluated the protocol for data access control using the scenarios from Section 4.5.2. The first scenario, the smart building scenario, describes that the building is on fire, and there exists a data access control rule that grants access to information of people's location for a fireman in emergency situations of fire.

$$\Delta_{\texttt{interface}} = \left\{ \begin{array}{l} \texttt{inf\_category(location(v1), end\_user\_inf)} \\ \texttt{role(nick, fireman)} \\ \texttt{constr(role(fireman), fireman)} \\ \texttt{satisfies(nick, role(fireman))} \\ \texttt{emrg\_rule(fire, access\_rule(fireman, end\_user\_inf))} \\ \texttt{fire} \end{array} \right\}$$

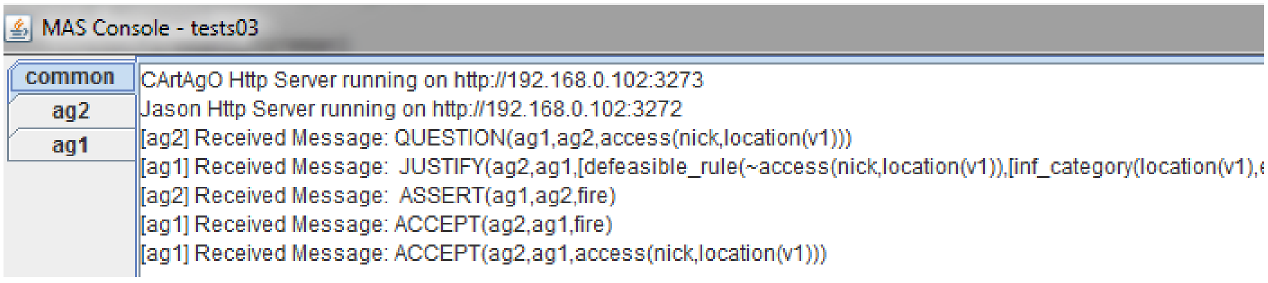


```
CArtAgO Http Server running on http://192.168.0.102:3273
Jason Http Server running on http://192.168.0.102:3272
[ag2] Received Message: QUESTION(ag1,ag2,access(nick,location(v1)))
[ag1] Received Message: ACCEPT(ag2,ag1,access(nick,location(v1)))
```

Figure 5.4 – First Example – Scenario 1

Considering the knowledge base $\Delta_{\texttt{interface}}$ (the interface agent ag_2), we have the output shown in Figure 5.4 from our implementation. In this first example, *Nick* (agent $ag1$ in our output), is a fireman, and it requests access to the information `location(v1)`, representing the location of a person inside the building. Considering that the interface agent knows that Nick is a fireman and it is an emergency situation of fire, it provides access to the information.

In the second example, we assume that the interface agent does not know that it is an emergency situation of fire, thus, when Nick requests access to that information, the interface agent first denies the access, justifying that nick has no access to that information. As Nick has been correctly identified as a fireman, which could be identified in the support of the argument used by the interface agent, it provides the information that it is an emergency situation of `fire` to the interface agent. With this new information, the interface agent grants access to that information. The output of our implementation is shown in Figure 5.5.



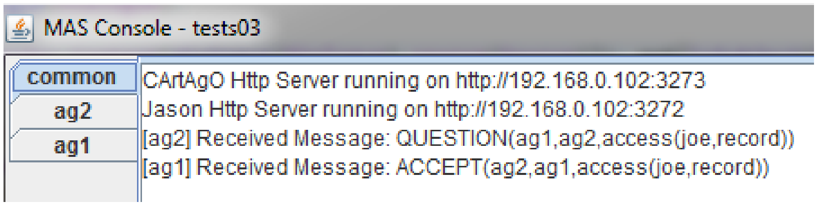
```
MAS Console - tests03
common  CArtAgO Http Server running on http://192.168.0.102:3273
ag2     Jason Http Server running on http://192.168.0.102:3272
ag1     [ag2] Received Message: QUESTION(ag1,ag2,access(nick,location(v1)))
        [ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(nick,location(v1)),[inf_category(location(v1),
        [ag2] Received Message:  ASSERT(ag1,ag2,fire)
        [ag1] Received Message: ACCEPT(ag2,ag1,fire)
        [ag1] Received Message: ACCEPT(ag2,ag1,access(nick,location(v1)))
```

Figure 5.5 – Second Example – Scenario 1

The second scenario, the travelling patient, describes that a person from a hospital requests access to the record of a patient from another hospital.

$$\Delta_{\texttt{interface}} = \left\{ \begin{array}{l} \texttt{constr(role(family\_doctor), family\_doctor)} \\ \texttt{constr(role(doctor), doctor)} \\ \texttt{inf\_category(record, patient\_data)} \\ \texttt{access\_rule(family\_doctor, patient\_data)} \\ \texttt{role(joe, doctor)} \\ \texttt{emrg\_rule(unc, access\_rule(doctor, patient\_data))} \\ \texttt{unc} \end{array} \right\}$$

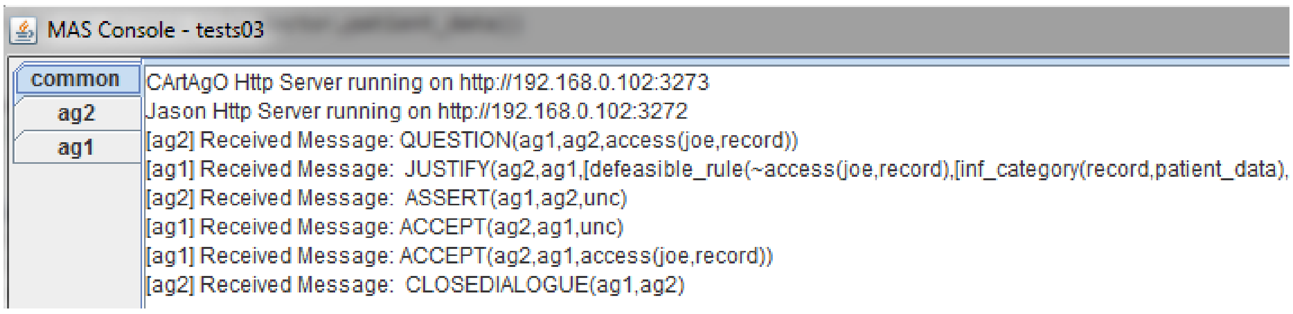
In the first example, the interface agent has all knowledge about the situation, i.e., it knows that the patient is unconscious, thus, when a doctor named Joe requests the information, the interface agent provides it. The output of our implementation is shown in Figure 5.6.



```
MAS Console - tests03
common  CArtAgO Http Server running on http://192.168.0.102:3273
ag2     Jason Http Server running on http://192.168.0.102:3272
ag1     [ag2] Received Message: QUESTION(ag1,ag2,access(joe,record))
        [ag1] Received Message: ACCEPT(ag2,ag1,access(joe,record))
```

Figure 5.6 – First Example – Scenario 2

In the second example, we simulate a situation in which the interface agent does not know that the patient is unconscious, and it has correctly categorised Joe as a doctor. Thus, first the interface agent denies access to that information, and considering that Joe (i.e., $ag1$) has been correctly categorised, it communicates the emergency situation in which the patient is unconscious. With this new information, the interface agent grants access to that information. The output from our implementation is shown in Figure 5.7.



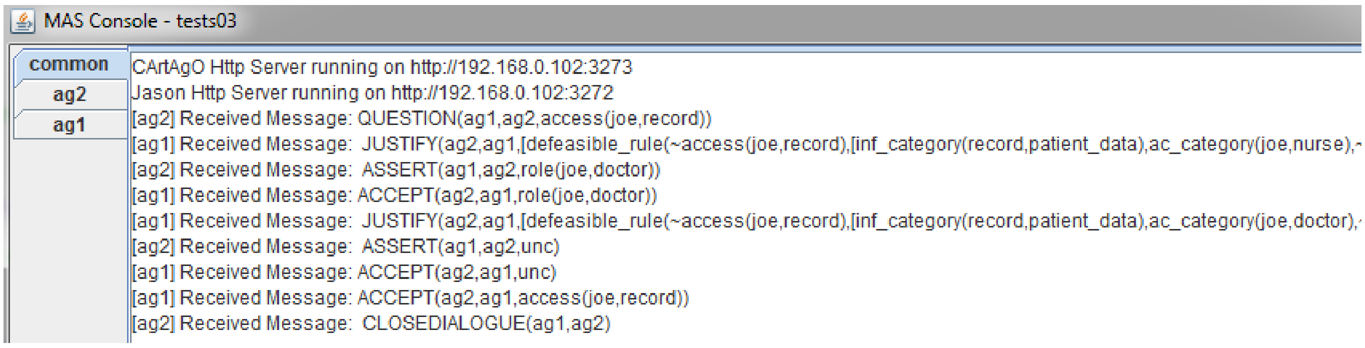
MAS Console - tests03

common
ag2
ag1

```
CArtAgO Http Server running on http://192.168.0.102:3273
Jason Http Server running on http://192.168.0.102:3272
[ag2] Received Message: QUESTION(ag1,ag2,access(joe,record))
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_category(record,patient_data),
[ag2] Received Message:  ASSERT(ag1,ag2,unc)
[ag1] Received Message: ACCEPT(ag2,ag1,unc)
[ag1] Received Message: ACCEPT(ag2,ag1,access(joe,record))
[ag2] Received Message:  CLOSEDIALOGUE(ag1,ag2)
```

Figure 5.7 – Second Example – Scenario 2

In the third example, we assume that the interface agent (i.e., *ag2*) receives the request and categorises Joe as a `nurse`. Thus, it denies access to the information, justifying that nurses have no access to the record of patients. When Joe receives this argument, it identifies that it has been wrongly categorised, providing its correct role in the system it belongs to. Even though the interface agent denies access, but now categorises Joe as a doctor. Thus, Joe informs the interface agent that the patient is unconscious, and the interface agent grants access to the information. The output from our implementation is shown in Figure 5.8.



MAS Console - tests03

common
ag2
ag1

```
CArtAgO Http Server running on http://192.168.0.102:3273
Jason Http Server running on http://192.168.0.102:3272
[ag2] Received Message: QUESTION(ag1,ag2,access(joe,record))
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_category(record,patient_data),ac_category(joe,nurse),~
[ag2] Received Message:  ASSERT(ag1,ag2,role(joe,doctor))
[ag1] Received Message: ACCEPT(ag2,ag1,role(joe,doctor))
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_category(record,patient_data),ac_category(joe,doctor),~
[ag2] Received Message:  ASSERT(ag1,ag2,unc)
[ag1] Received Message: ACCEPT(ag2,ag1,unc)
[ag1] Received Message: ACCEPT(ag2,ag1,access(joe,record))
[ag2] Received Message:  CLOSEDIALOGUE(ag1,ag2)
```

Figure 5.8 – Third Example – Scenario 2

5.6 An Approach for Enthymemes in Multi-Agent Systems

Enthymemes are arguments in which one or more statements (which are part of the argument) are not explicitly stated, i.e., they are arguments with “missing premises” or even “missing conclusions” [156]. They are more realistic arguments, in the sense that real-world arguments (i.e., arguments presented by humans) usually do not have enough explicitly presented premises for the entailment of the claim [23]. Further, some authors describe that arguments can be naturally considered enthymemes, because they are not based on strict consequence, in which the conclusion follows the premises in a strict way, but based on *enthymematic consequence* [60]. That is, arguments are based on reasoning patterns (argumentation schemes) in which only part of the knowledge necessary to instantiate an acceptable instance of that argumentation scheme is explicitly represented in the argument, missing, for example, information that is identified through the critical questions.

Using enthymemes in human activities has widely known benefits, allowing economy, efficiency, and efficacy [90]. For example, the authors in [90] note that:

1. being short it can easily be followed by the audience;
2. starting from what is known and already taken for granted is important, because the addressee will, in consequence, be well-disposed towards what is said;
3. omitting premises is also justified by reasons of economy, where if something is well known it is unnecessary to repeat it;
4. omitting the conclusion has advantages in terms of economy, but might also be beneficial to efficiency because if the audience comes to the conclusion by itself, its acceptance is more easily acquired; and
5. allowing members of the audience to come to a conclusion by themselves might be more respectful towards them.

Besides the use of enthymemes characterising more rational and "intelligent" agents, bringing them closer to human reasoning and communication, they also could bring some computational benefits for multi-agent systems, as we discuss in this section.

When agents use enthymemes, the common knowledge is removed from the original argument by the proponent of such an argument (such process is called *encoding* in [23]), where it is assumed that all information removed from the original argument is known by the recipients and, therefore, they are able to reconstruct (or to *decode* [23]) such an argument, understanding exactly what meant to be communicated. If both the proponent and recipients use the same common knowledge, then this process is straightforward [67]. However, normally, this is not the case, and the recipients generate an approximation of such an argument in the reconstruction process, given the disparities between the different views on what constitutes common knowledge [67].

In our framework for argumentation scheme in multi-agent systems, we can implement enthymeme-based communication, guaranteeing that all agents, which are involved in an argumentation-based dialogue, will have the same understanding when using enthymemes, by means of guiding the reconstruction of arguments by the recipients using argumentation schemes and available organisational knowledge. Also, recall that we have an infrastructure that ensures that all agents will have the same perception on what constitutes common knowledge[9] (i.e., organisational information and argumentation schemes).

Definition 23 (Enthymeme). *Let* $\langle S, c\rangle^{\theta}_{\mathtt{sn}}$ *be an acceptable argument to agent* $\mathtt{ag_i}$. *An enthymeme for* $\langle S, c\rangle^{\theta}_{\mathtt{sn}}$ *is a tuple* $\langle S', c\rangle^{\theta}_{\mathtt{sn}}$, *where* $S' \subset S$ *and* $S' \subseteq (\Delta_{\mathtt{ag_i}} \setminus (\Delta_{\mathtt{Org}} \cup \Delta_{\mathtt{AS}}))$.

[9]In this section, we use $\Delta_{\mathtt{Org}}$ and $\Delta_{\mathtt{AS}}$ in order to represent the organisational information and the argumentation schemes available for agents, respectively.

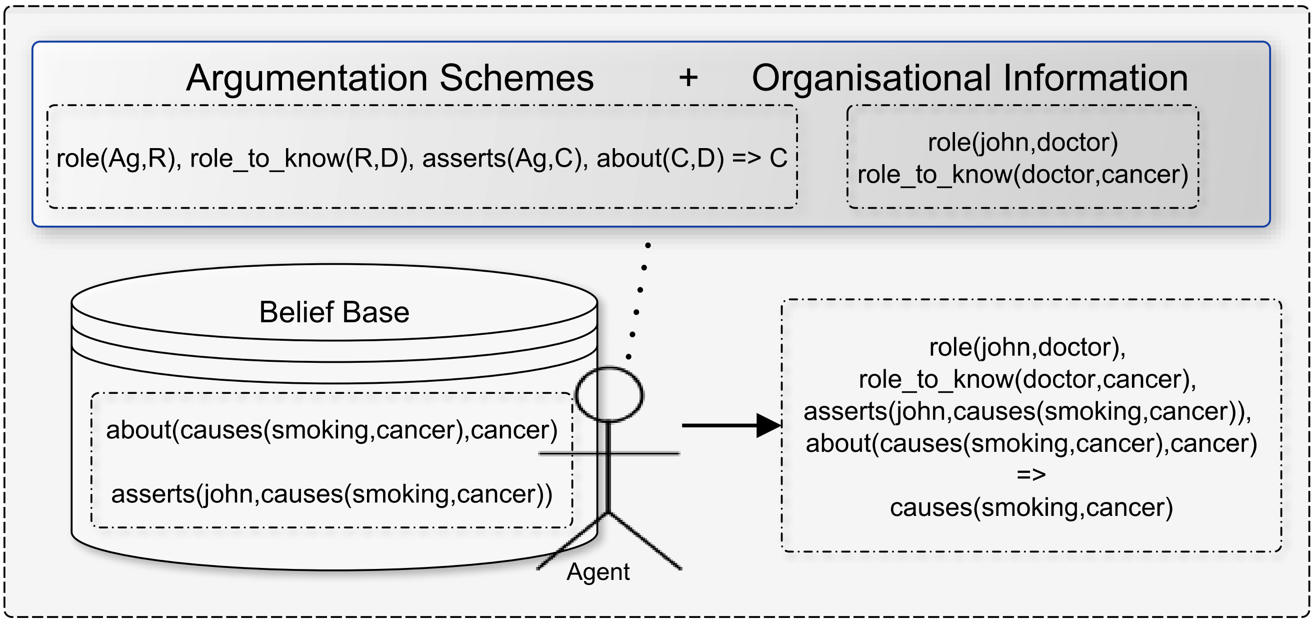

Argumentation Schemes + Organisational Information
role(Ag,R), role_to_know(R,D), asserts(Ag,C), about(C,D) => C
role(john,doctor)
role_to_know(doctor,cancer)
Belief Base
about(causes(smoking,cancer),cancer)
asserts(john,causes(smoking,cancer))
Agent
role(john,doctor),
role_to_know(doctor,cancer),
asserts(john,causes(smoking,cancer)),
about(causes(smoking,cancer),cancer)
=>
causes(smoking,cancer)

Figure 5.9 – Constructing arguments using argumentation schemes and organisational information.

Returning to our example in which an agent instantiate an argument from the argumentation scheme *role to know* based on the assertion of a doctor named `john`. That argument contains, in its support, all premises and inference rules needed to entail the conclusion, and it could be constructed as:

$\langle$ { `defeasible_rule(causes(smoking,cancer),`
`[asserts(john,causes(smoking,cancer)),role(john,doctor),`
`role_to_know(doctor,cancer), about(causes(smoking,cancer),cancer)]),`
`asserts(john,causes(smoking,cancer)),role(john,doctor),`
`role_to_know(doctor,cancer),about(causes(smoking,cancer),cancer)`},
`causes(smoking,cancer)` $\rangle^{\theta}_{\texttt{[as(as4rk)]}}$

The corresponding enthymeme has the following format (considering the common knowledge presented in Figure 5.9):

$\langle$ { `asserts(john,causes(smoking,cancer)),`
`about(causes(smoking,cancer),cancer)`},
`causes(smoking,cancer)` $\rangle^{\theta}_{\texttt{[as(as4rk)]}}$

Based on the label [as(as4rk)] and the most general unifier θ, an agent receiving the enthymeme is able to identify the missing premises, understanding which information the sender agent has used to conclude `causes(smoking,cancer)`, given that all missing premises are organisational information and, therefore, common knowledge. Also, based on the argumentation scheme annotated in the argument, the receiver agent is able to identify that information that is implicit in the argument, pointed out by the critical questions of that scheme.

5.6.1 Semantics for Speech-acts using Enthymemes

We formalise the construction and reconstruction of enthymemes (i.e., the encoding and decoding processes) by agents, associated with the speech act used by agents in the interactions. Therefore, we extend the operational semantics for the speech acts described in this chapter, including the encoding and decoding processes mentioned, which are associated with receiving and sending messages. This semantics has been partially published in [103]. The operational semantics extends the previous semantics in the agent configurations $\langle ag, C, M, T, s\rangle$, thus we only write the components necessary to describe those processes.

In the interest of readability, we adopt the following notational conventions in our semantic rules:

- We write: (i) $\mathtt{b[s(}id\mathtt{)]}$ to identify the origin of a belief, where id is an agent identifier (`s` is an abbreviation for `source`); (ii) $[\mathtt{dec(sn_i)}]$ to identify information that was decoded from an enthymeme, guided by the argumentation scheme $\mathtt{sn_i}$; and (iii) $\mathtt{b[as(sn_i,}\theta\mathtt{)]}$ to identify the argumentation scheme and the unifier used by the other agent in the received move.
- We use a function `prem()` which returns all premises in the support of the argument, e.g., $\mathtt{prem(}S\mathtt{)}$ returns all premises in the support of the argument $\langle S, c\rangle$.

5.6.2 Semantics for Sending Messages

In this section, we present the formal semantics for agents sending the speech acts, in which we formalise the process for encoding arguments into enthymemes.

$$\frac{\begin{array}{c} T_\iota = i[head \leftarrow .\mathtt{send}(id, \mathtt{assert}, \langle S, c\rangle^{\theta}_{\mathtt{sn_i}});h] \\ S \subset \Delta_{\mathtt{ag}} \qquad \Delta_{\mathtt{ag}} \models c \qquad \langle \mathtt{sn_i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{\mathtt{AS}} \\ \forall p \in \mathcal{P}, p\theta \in \mathtt{prem}(S) \qquad c = \mathcal{C}\theta \end{array}}{\langle ag, C, M, T, \mathsf{ExecInt}\rangle \longrightarrow_{AS} \langle ag, C', M', T, \mathsf{ProcMsg}\rangle} \ (\mathbf{ExActSndAssert})$$

where:

$$\begin{array}{lcl} M'_{Out} & = & M_{Out} \cup \{\langle mid, id, \mathtt{assert}, \langle S', c\rangle^{\theta}_{\mathtt{sn_i}}\rangle\} \\ & & \text{with } S' = S \setminus (S \cap (\Delta_{\mathtt{Org}} \cup \Delta_{\mathtt{AS}})) \\ C'_I & = & (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\} \end{array}$$

Sending an `assert` or `justify` message: when an agent executes the internal action for sending a message with the performative `assert` or `justify`, the agent needs to have an acceptable argument for that particular conclusion[10], which was drawn using the argumentation schemes $\mathtt{sn_i}$.

[10]While we use, in this work, the *thoughtful* attitude [116, 114], other agent attitudes could be used just as well.

Such argument is encoded into an enthymeme $\langle S', c\rangle^{\theta}_{\mathtt{sn_i}}$, where all common knowledge is removed from the argument support, i.e., in this work, the organisational knowledge $\Delta_{\mathtt{Org}}$ and the contents of the argumentation scheme used $\Delta_{\mathtt{AS}}$ are removed. The corresponding message is posted in the agent mailbox and the current agent intention is updated, removing the internal action, given that its execution is completed. Here, we only present the **ExActSndAssert** semantics rule, given that the semantic rule for the performative `justify` has a similar formalisation.

5.6.3 Semantics for Receiving Messages

In this section, we present the formal semantics for agents receiving the speech acts. Furthermore, the semantics formalise the process of decoding enthymemes into arguments.

Receiving an `assert` **or** `justify` **message**: when an agent selects an `assert` or `justify` message from its mailbox, the message is removed from the agent's mailbox, the agent's belief base is updated with all information contained in the support of the intended argument, annotating clearly which formulæ have been received from the sender and which have been decoded. Further, the agent's belief base is updated with the information that the sender has asserted c using the argumentation scheme $\mathtt{sn_i}$ grounded by θ, and the respective event is generated for that.

$$\frac{\begin{array}{c} S_M(M_{In}) = \langle mid, sid, \mathtt{assert}, \langle S, c\rangle^{\theta}_{\mathtt{sn_i}}\rangle \\ \langle \mathtt{sn_i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{\mathtt{AS}} \qquad \forall p\theta \in S, p \in \mathcal{P} \qquad c = \mathcal{C}\theta \end{array}}{\langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow_{AS} \langle ag', C', M', T, \mathsf{ExecInt}\rangle} \quad \textbf{(Assert)}$$

where:

$$\begin{array}{lcl} M'_{In} & = & M_{In} \setminus \{\langle mid, sid, \mathtt{assert}, \langle S, c\rangle^{\theta}_{\mathtt{sn_i}}\rangle\} \\ ag'_{bs} & = & ag_{bs} \cup \{p[s(sid)]\theta \mid \text{for all } p \in \mathcal{P} \text{ and } p\theta \in S\} \cup \\ & & \{p[s(sid), dec(\mathtt{sn_i})]\theta \mid \text{for all } p \in \mathcal{P} \text{ and } p\theta \notin S\} \cup \\ & & \{asserts(sid, c)[as(\mathtt{sn_i}, \theta)]\} \\ C'_E & = & C_E \cup \{\langle +asserts(sid, c)[as(\mathtt{sn_i}, \theta)], \mathsf{T}\rangle\} \end{array}$$

It is important to mention that, when the agent receives an assert or justify message, the content is an enthymeme. Thus, the enthymeme is decoded into the original sender's argument, guided by the argumentation scheme, and the agent that receives the enthymeme updates its belief base with all premises in the intended argument, including the sender as the source for all that information, but also identifying the decoded premises with $\mathtt{dec(sn_i)}$.

Implementation in Jason Agents

We have implemented an interface, as we implemented the updates to the agents' commitment stores, which encode the arguments to enthymemes and *vice versa*. This interface uses a Jason module we implemented, in which the following queries are available:

- `enthymemetisation(Arg, Enthymeme, AS, Unifier)`: which receives the intended argument `Arg`, and returns the corresponding enthymeme `Enthymeme`, the argumentation scheme used to instantiate that argument `AS`, and the unifier function used to instantiate that argument from that argumentation scheme `Unifier`. This query uses a predicate `annot2remove(`$<$ `List` $>$`)`, in which a list of annotation identify the predicates that should be removed from the intended argument during the process of enthymemetisation (to encode an argument to an enthymeme). In our examples, we have used the following list of annotations `annot2remove([source(org), as(_)])`, meaning that all information from the organisation, annotated with `source(org)`, as well as all argumentation scheme rules, annotated with `as(_)`, will be removed from the intended argument.

- `decodification(Enthymeme, AS, Unifier, ArgD)`: which receives an enthymeme `Enthymeme`, an argumentation scheme `AS`, and a unifier function `Unifier`, and returns the intended argument `ArgD` properly annotated.

Although we only remove organisation knowledge and argumentation scheme from the intended arguments in our examples, it can be easily extended through a different list of annotated predicates to remove, defined through the predicate `annot2remove(`$<$ `List` $>$`)`. For example, it could be interesting to remove other information an agent has learned to be shared knowledge. Thus, a likely direction for this work could be combining our work on Theory of Mind (ToM) [95, 96, 134] with our approach for enthymemes.

Example

Consider a short dialogue between a nurse and two patients in a hospital scenario using the argumentation scheme from role to know `as4rk`. Imagine that the nurse is supposed to explain to two patients, *Alice* and *Bob*, who were diagnosed with cancer, how they probably got the illness[11]. Following the protocol introduced for persuasion, the nurse executes an `assert` move explaining that smoking causes cancer, in which the following enthymeme is resulting from the **ExActSndAssert** rule:

$\langle$`{asserts(john,causes(smoking,cancer)),`
`about(causes(smoking,cancer),cancer)]}, causes(smoking,cancer)`$\rangle^{\theta}_{[\mathtt{as(as4rk)}]}$

Based on the common knowledge that `john` is a doctor and doctors are experts in such domains (see Figure 5.9), both patient accept that "*smoking causes cancer*". While we have a short and human-like communication, the understanding of the arguments is the same than communicating it entirely.

[11]It is important to mention that we assume the nurse was asked to explain the doctor's diagnosis; we do not assume that the nurse made the diagnosis themselves.

5.6.4 Properties

In this section, we describe the main properties of our approach for enthymemes, namely: (i) both agents, the sender and the receiver, will have the same understating of what has been uttered, ensuring that arguments will not lose their meaning; (ii) agents will exchange shorter content, given that arguments are encoded into enthymemes. (iii) enthymemes can be built to an entire audience, and not only to a particular recipient. The properties presented here are important aspects for multi-agent applications that are run distributed over a network, possibly using mobile devices, etc.

Proposition 1. *Both agents, sender and receiver, will have the same understanding of the uttered arguments.*

Proof (sketch). The architecture proposed in this work guarantees that an argument is decoded exactly as the intended argument, which was previously encoded into an enthymeme. This process is guaranteed by the semantic rules presented in the previous section, especially for sending and receiving the `assert` and `justify` moves. The process uses the general unifier θ and the annotation of the reasoning pattern used, i.e., the argumentation scheme used to draw up that particular conclusion. Thus, it is guaranteed that the same reasoning pattern will be used to decode the enthymeme into the intended argument and that all variables will be correctly instantiated, based on θ. □

Proposition 2. *Agents will exchange less content using our approach.*

Proof (sketch). When agents utter new arguments, i.e., they execute an `assert` or `justify` move, the argument is encoded into an enthymeme. This means that all common knowledge (in this work, specifically organisation-related knowledge $\Delta_{\texttt{Org}}$ and the argumentation schemes $\Delta_{\texttt{AS}}$) is removed from the intended argument (the removed information is never transmitted from sender to receiver). Even for an argument which was instantiated without the use of any organisational knowledge, still shorter content will be uttered than the intended argument, given that the inference rule used to draw that particular conclusion (from $\Delta_{\texttt{AS}}$) will be removed from the original argument, as it can be retrieved by the receiver from the scheme. □

Proposition 3. *Enthymemes can be built to an entire audience within the same organisation.*

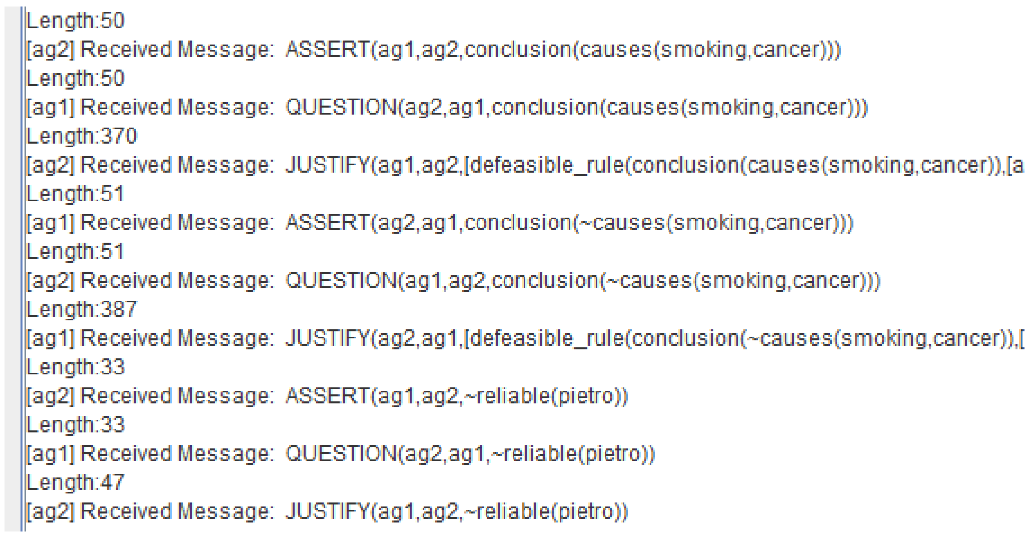
Proof (sketch). Considering that all agents belong to the same organisation, argumentation schemes are shared by all agent in that organisation, and all agents are aware of the organisational structure, i.e., all agents share $\Delta_{\texttt{Org}}$ and $\Delta_{\texttt{AS}}$, an enthymeme can be built to an entire audience, and all agents will have the same understanding of that uttered argument. The proof follows the Proposition 1, considering that the information removed from the intended argument is known by all agents in that organisation and all agents will use the same argumentation scheme (also known by all agents) to reconstruct the intended argument. □

5.6.5 Evaluation

We ran some experiments to evaluate our approach for enthymeme in multi-agent systems. Note that the Propositions 1 and 3 are easy to follow, given our formal framework for enthymemes in multi-agent systems, and the implementation that follows this formal definition. Also, in Proposition 2, it is easy to note that agents will always remove the argumentation scheme they used to instantiate an argument when communicating, adding only a reference to this argumentation scheme, which is shared by all agents. Besides omitting the argumentation schemes, agents will also omit organisational knowledge, but how much organisation knowledge agents are using in their argument will depend on the application domain and scenarios of argumentation.

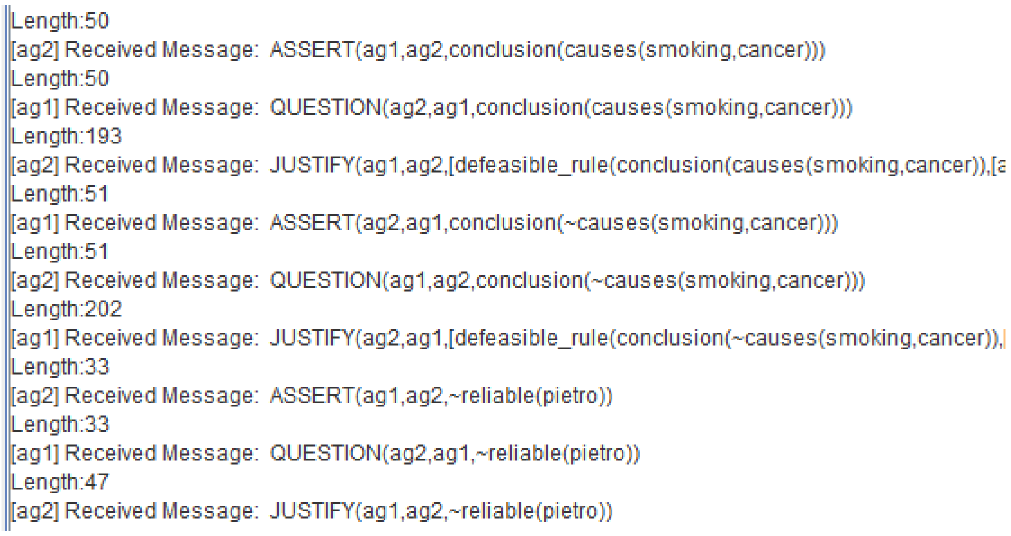
In the experiments for argumentation-based dialogue using enthymeme, we extended the agent architecture in Jason in order to show the length of messages communicated by agents. Using this extended architecture we are able to compare dialogues, showing how much content is reduced when agents communicate enthymemes instead of arguments.

In Figure 5.10 we show an output of the persuasion dialogue presented in Section 5.5.2, including the length of messages. In Figure 5.11 we show the same persuasion dialogue, but now using enthymemes.



```
Length:50
[ag2] Received Message:  ASSERT(ag1,ag2,conclusion(causes(smoking,cancer)))
Length:50
[ag1] Received Message:  QUESTION(ag2,ag1,conclusion(causes(smoking,cancer)))
Length:370
[ag2] Received Message:  JUSTIFY(ag1,ag2,[defeasible_rule(conclusion(causes(smoking,cancer)),[a
Length:51
[ag1] Received Message:  ASSERT(ag2,ag1,conclusion(~causes(smoking,cancer)))
Length:51
[ag2] Received Message:  QUESTION(ag1,ag2,conclusion(~causes(smoking,cancer)))
Length:387
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(conclusion(~causes(smoking,cancer)),[
Length:33
[ag2] Received Message:  ASSERT(ag1,ag2,~reliable(pietro))
Length:33
[ag1] Received Message:  QUESTION(ag2,ag1,~reliable(pietro))
Length:47
[ag2] Received Message:  JUSTIFY(ag1,ag2,~reliable(pietro))
```

Figure 5.10 – Length of messages using arguments

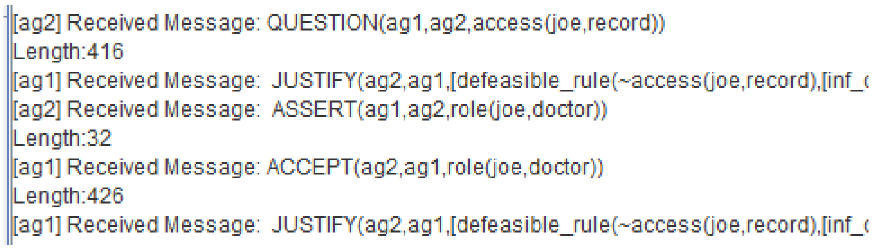


```
Length:50
[ag2] Received Message:  ASSERT(ag1,ag2,conclusion(causes(smoking,cancer)))
Length:50
[ag1] Received Message:  QUESTION(ag2,ag1,conclusion(causes(smoking,cancer)))
Length:193
[ag2] Received Message:  JUSTIFY(ag1,ag2,[defeasible_rule(conclusion(causes(smoking,cancer)),[a
Length:51
[ag1] Received Message:  ASSERT(ag2,ag1,conclusion(~causes(smoking,cancer)))
Length:51
[ag2] Received Message:  QUESTION(ag1,ag2,conclusion(~causes(smoking,cancer)))
Length:202
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(conclusion(~causes(smoking,cancer)),
Length:33
[ag2] Received Message:  ASSERT(ag1,ag2,~reliable(pietro))
Length:33
[ag1] Received Message:  QUESTION(ag2,ag1,~reliable(pietro))
Length:47
[ag2] Received Message:  JUSTIFY(ag1,ag2,~reliable(pietro))
```

Figure 5.11 – Length of messages using enthymemes

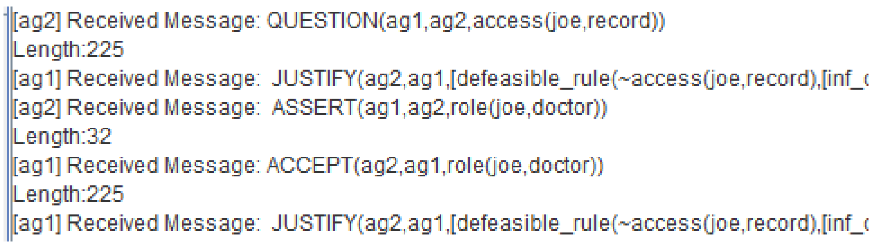
In this experiment, agents reduce the length of messages, when communication enthymeme, in an average of 48% (45% the first argument, and 51% the second argument). Note that two premises of these arguments are organisational knowledge.

In Figure 5.12 we show an output of the dialogue for data access control presented in Section 5.5.3, including the length of messages. In Figure 5.13 we show the same dialogue, but now using enthymemes.



```
[ag2] Received Message: QUESTION(ag1,ag2,access(joe,record))
Length:416
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_
[ag2] Received Message:  ASSERT(ag1,ag2,role(joe,doctor))
Length:32
[ag1] Received Message: ACCEPT(ag2,ag1,role(joe,doctor))
Length:426
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_
```

Figure 5.12 – Length of messages using arguments



```
[ag2] Received Message: QUESTION(ag1,ag2,access(joe,record))
Length:225
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_
[ag2] Received Message:  ASSERT(ag1,ag2,role(joe,doctor))
Length:32
[ag1] Received Message: ACCEPT(ag2,ag1,role(joe,doctor))
Length:225
[ag1] Received Message:  JUSTIFY(ag2,ag1,[defeasible_rule(~access(joe,record),[inf_
```

Figure 5.13 – Length of messages using enthymemes

In this experiment, agents reduce the message size, when communication enthymeme, in an average of 47% (46% the first argument, and 48% the second argument).

5.7 Final Remarks

In this chapter, we presented an argumentation framework for argumentation-based dialogues in multi-agent systems based on the argumentation schemes structure we presented in Chapter 3, combined with the argumentation-based reasoning mechanism we presented in Chapter 4. First we presented the formal semantics for performatives commonly used in argumentation-based dialogue we have published in [108, 111]. Those semantics rules formalise, in a precise way, the updates agents execute in their mental attitudes according to the performative received and/or sent. After, we have presented the artifacts we developed to support argumentation-based dialogue, which allowed to implement the semantics rules in a multi-agent system platform. We also argue that such artifacts facilitate programmers to implement different argumentation-based protocols in multi-agent systems, given the functionalities they provide. After, we have discussed the role of argumentation schemes in argumentation-based dialogues, in which argumentation schemes enable agents to use extra moves in argumentation-based dialogues based on that information that is implicit in the arguments used by agents. That information becomes explicit to agents when the argumentation scheme used to instantiate that argument is identified.

Considering the formal semantics proposed in this chapter, the artifacts to support argumentation-based dialogue in multi-agent systems, and our considerations about argumentation schemes in multi-agent systems, we evaluated our framework defining two different protocol that takes into account the proposed structure for arguments, including the references to shared argumentation schemes. We formalise the protocols using a formal framework [110], we show how the formal protocol can be implemented towards Jason plans, and we evaluated different outputs for each protocol, considering different agents configurations.

Finally, we present an approach for enthymemes (shorted and rational arguments) for multi-agent systems. In our approach, agents exchange only the information needed to ensure that both sides, the proponent and recipient(s), will have the same understanding of the uttered arguments. Our approach allows agents to exchange fewer messages with shorter content, which could be very useful both for making argumentation-based communication more efficient as well as for avoiding network overload. Improving the efficiency of argumentation-based communication could benefit all applications that use argumentation techniques, and decreasing the network usage could benefit all applications that use constrained networks, e.g., applications which use the mobile network such as [136, 74]. In this version of our approach for enthymemes in multi-agent systems, we allow agents to omit organizational knowledge only (but other approaches can be easily incorporated). Thus, note that considering organisational information and argumentation schemes as common knowledge, our approach allows agents to construct appropriate enthymemes not only for specific agents but also to an entire “audience”, when the audience belongs to the same organisation.

Also, we noted in the literature that considering argumentation schemes in multi-agent systems from a practical point of view seems as yet an underdeveloped research topic. Here, we described how we used argumentation schemes integrated with multi-agent platforms, and how they are used by agents to instantiate arguments and to refer to the critical questions for those arguments during argumentation-based dialogues.

6. RELATED WORK

In this chapter, we discuss the main work related to our argumentation-based framework for argumentation scheme in multi-agent systems. As can be noted, the main contribution of our work is a formal and implemented framework for argumentation schemes in multi-agent systems. On the one hand, to the best of our knowledge, our approach is the first approach to propose such a framework. On the other hand, there are a few works that consider (or propose) argumentation schemes in multi-agent systems. In Section 6.1 we will detail each one of them, relating them to our framework.

Further, an interesting direction in our work was the research regarding enthymemes, which seems a recent topic of research in multi-agent systems. Given its potential to bring agent communication closes to human-like communication, we dedicate Section 6.2 to discuss the related work about this topic, and in which points our work contributes to the state-of-art in enthymemes-based communication frameworks.

Finally, in Section 6.3, we discuss the related work concern on the problem of data access control in multi-agent systems (or general applications) using argumentation-based techniques. We dedicate that section to this topic, given that we gave special interest to this kind of application when evaluating our framework. As described, it seems that our work is the most general approach for data access control using argumentation-based interface agents.

6.1 Argumentation Schemes in Multi-Agent Systems

Recently, Parson *et al.* [113] present a set of argumentation schemes for reasoning about trust. The authors argue that trust is a natural mechanism by which an autonomous party can deal with the inherent uncertainty regarding the behaviours of other parties and the uncertainty in the information it shares with those parties. Emphasising that trust is crucial in any decentralised system. The schemes presented in [113] are abstract patterns of reasoning which can be applying in multiple situations, geared toward trust. The argumentation schemes are the following: *trust from direct experience*, *trust from indirect experience*, *trust from expert opinion*, *trust from authority*, *trust from reputation*, *trust from moral nature*, *trust from social standing*, *trust from majority behaviour*, *trust from prudence*, *trust because of pragmatism.* For example, the argumentation scheme of *Trust from Authority* [113] specifies that *if B is in a position of authority, then A may trust in B.* The critical questions of this scheme are: Is B really in position of authority? Is B's authority relevant in this case?, etc. Exist an interesting link between our research and the work of Parsons *et al.* [113]. For example, the argumentation scheme of *Trust from Authority* discussed, is very interesting to our framework. We can extend this argument to specifies that A can believe that if B says something this information probably is true, because the position of authority of B. We can extend the critical questions as well, questioning if the role of the agent B (considering the organisation) is relevant in the field where the information is contained, and thus can be considered true. The link between argumentation schemes and the organisation of a multi-agent system is a very interesting topic of research. While we made a few links between multi-agent systems organisation and argumentation schemes, there are many other argumentation schemes in the literature in which this link also is possible, and [113] is one of these works.

The, also recent, work of Toniolo *et al.* [147, 146] proposes an argumentation-based model for deliberation dialogues based on argumentation schemes. The authors argue that their model facilitates agreements about joint plans by enriching the quality of the dialogue through the exchange of relevant information about plan commitments and norms. The work [147] proposes three argumentation schemes that agents can use during the discussion according to issues of practical reasoning such as concurrent actions (*arguments for concurrent actions*), causality among actions (*arguments for plan constraints*) and norms (*arguments for norms.*). Some critical questions presented are: Is the action possible according to concurrent action in the plan? Is the action possible according to causal plan constraints? Is there any norm which regulates actions or states of the world? For example, in *arguments for norms*, if a norm obliges the execution of an action, the agent can use the attack that this action should be executed. The work of Toniolo *et al.* [147] is related to the execution of actions in multi-agent systems. Although our focus is not the execution of actions, the consideration of norms is relevant to our research, where the organisation model specifies which is allowed and prohibited to be done by the agents. This work is another example specifying argumentation schemes that can be linked to the multi-agent system organisation.

In [145], Toniolo *et al.* propose the argumentation schemes which present the essential elements of provenance that warrant the credibility of the information. The authors argue that the introduction of schemes about provenance facilitates the decision-making process by providing a rational method to assess the credibility of a piece of information and to resolve conflicting information. The argumentation scheme for provenance extends the argumentation scheme of position to know [156]. This work [145] also is relevant to our research, where, similar to it, we have extended the argumentation schemes of position to know considering the role of the agent which the information is provenance from. Also, the authority link between the agent specified by the organisational model, combined with the argumentation scheme from [145] could be considered in future work.

In [131], the authors describe that "argumentation schemes are patterns of non-deductive reasoning that have long been studied in argumentation theory, and have more recently been identified in computational domains including multi-agent systems as holding the potential for significant improvements in reasoning and communication abilities". The work demonstrates that (i) individual agents can reasoning about and develop arguments that employ schemes, and (ii) that communication structures can be built up around those schemes. Two points refer possible implementation of argumentation schemes following [131]: (i) the representation and manipulation of schemes by one agent; (ii) the use of this representation in multi-agent cases, where this representation can be explored in communication design. The authors suggest ARAUCARIA [129, 130] as a possible manner to represent arguments, where ARAUCARIA uses AML (Argument Markup Language) as the language to represent arguments (AML is an underlying language XML). The authors, further, argue that implementing scheme-based communication situated in a multi-agent system is currently a work in progress. Another claim from [131], is that the way in which particular schemes are judged is a feature of the community or society in which that agent resides (demonstrating a close analogy to human communities). The claims found in [131] demonstrate that the consideration of an organisational model linked to the specification of argumentation schemes is a promising research area, as well as the work described above confirms these claims. Also, it has inspired us in the modelling of our framework, in which we looked to a formal argumentation scheme structure in order to model our framework.

In [55], the authors claim that Dung's work in abstract argumentation does not fit a wide class of arguments used by human argumentation where pros and cons are balanced in order to choose among different options. Thus, the authors in [55] propose a formal model of structured argumentation which generalises Dung's abstract argumentation frameworks to also handle with the balancing of arguments. Differently from other work in structured argumentation, [55] generalises abstract argumentation frameworks, which could be simulated using structured argumentation. An interesting point in [55], is that the authors divide human argumentation into different types, including: (i) *practical argumentation*, comparing arguments on pros and cons to choose a course of action, including arguing about preconditions and effects of the actions; (ii) *theoretical argumentation*, constructing and comparing arguments to choose among alternative theories (choosing the more coherent theory); (iii) *factual argumentation*, arguing about whether or not some event occurred; etc. An important issue in [55], we emphasise, is that they are not interested in trying to find some way to model the approach proposed in Dung's methodology. In that work, argumentation schemes play the role of abstract structures for generating, validating and weighing arguments (however, in that paper, only the weighing function was presented, given that they were interested only in balancing arguments). In the framework, arguments are a tuple (s, P, c, u), with s the argumentation scheme instantiated by the argument, P the premises of the argument (a finite subset of the language), c the conclusion of the argument, and u the undercutter of the argument. The framework considers *issues* instead of a *contrary* binary relation, where issues is n-ary relation. Similar to [55], we also use argumentation schemes as abstract reasoning patterns, but in our work related to multi-agent systems instead. The framework proposed in [55] has similar concepts to our proposal for argumentation schemes in multi-agent systems, and such concepts are considered in our research. Although we have not considered weighting argument, as in [55], it is a likely direction to our research, combining our framework for argumentation schemes in multi-agent systems, and our previous work on using meta-information in argumentation-based reasoning [82, 83, 84, 85, 107].

In [165], the author describes that argumentation schemes could require a computational approach where agents use the components of a scheme to construct and present arguments and counterarguments, thus argumentation schemes could be useful in multi-agent systems. In that work, the author proposes a syntactical analysis for argumentation schemes, helping to clarify what is needed in order to provide denotations of the terms and predicates in a semantic model. Further, the author claims that, from a wide perspective, the analysis proposed begins to bridge the gap between the realisation of argumentation in natural language and their formal analysis. Following the analysis presented in [165], our work in representing argumentation schemes in agent-oriented programming languages could cover different levels of descriptions of argumentation schemes. For example, in an agent belief base, argumentation schemes are represented according to the agent-oriented programming language, in our case, Jason predicates, therefore they could be classified as *Labelled roles and Instantiated predicates* [165], while in the organisational model such argumentation schemes could be represented in XML. Furthermore, in [165] the author describes different kinds of argumentation schemes, which could play a part in our research, including argumentation schemes for arguing about facts, direct analysis, and reanalysis.

In [162], the author identified a property that captures the relationship between dialogue games and argumentation schemes called "scheme awareness". This property describes the degree to which a given game can exploit schemes within a dialogue and the game features that support this. The aim of that work is to provide a guide for designing dialogue games enabling them to better exploit argumentation schemes. The author argues that many sets of argumentation schemes can be used to annotate a specific analyse of argument structure, and in this context such argumentation schemes provide a mechanism for collecting, comparing, and evaluating instances of arguments, allowing yet to model a variety of reasoning methods making the computational reuse more receptive to argumentation. Further, the author argues that argumentation schemes can be used in relation to dialogue games as both: (i) at the development stage, providing a guide to developing new game rules; and (ii) at the deployment stage, providing relevant lines of argument for the player to explore, as well as to suggest appropriate responses to the others' positions, and to provide a facet of strategic information players may use to achieve their goals. Also, the author emphasises that while dialogue games have become a popular approach to structured interaction, for both agents-agents and agents-humans, the dialogue games available do not exploit the benefits of using argumentation schemes [162].

Further, the author in [162] suggests that the literature in argumentation schemes and dialogue games can be classified as follows:

- Games unable to utilise (i.e., represent and manipulate) argumentation schemes
- Games able to utilise a single argumentation scheme
- Games able to utilise multiple/arbitrary argumentation schemes

Also, the author describes that there is no game at the third level, considering multiple argumentation schemes. At the second level, in which the game is able to utilise a single argumentation scheme, the author describes some games that utilise the practical reasoning scheme [9]:

> In the current Circumstances, R, we should perform Action, A, to achieve New Circumstances, S, which will realise some goal, G, which will promote some value, V

Between such works, the author describes the PARMA protocol [10]. Another work is Toulmin Dialogue Game or TDG [14] that is explicitly based upon a specific argumentation scheme called Toulmin Argument Scheme [149]. The author describes that a game should support the expression of an argument as a single complex utterance. Regarding this point, the author describes that, even single move can present distinct forms, for example, the assert move could be used both to express an entire argument within a single locution or to express, individually or in combination, the constituent parts of an argument (conclusion, major and minor premise(s)). The author emphasises that the second option enables dialogue games to be specified in such a way that enables expressive, fine-grained, and more natural dialogues.

Furthermore, the author [162] proposes to extend the Dialogue Game Description Language (DGDL) [163] with meta-information in order to make clear the role of each part in the utterances made by agents. Thus, that author suggests to label content as an instance of an argument, to label the conclusion of an argument, to label the major and minor premises of an argument, to label content as an element of a type of argumentation scheme, and to label content as part of an instantiated argumentation scheme. Thus the author use a representation of "key:values" to label the content, e.g., assert("conclusion":"p", "major-premise":"q"), assert("conclusion":"p", "scheme-name":"expert-opinion"). The author argues that the aim of the paper is to explore the minimal set of requirements to enable the labelling of move content with argumentation schemes using meta-data. Aiming sufficient support for arguments and argumentation schemes in dialogue games.

Our work has similar directions to [162], in the sense that we allow agents to identify the different parts in an utterance through the semantics we introduced for speech acts used in argumentation-based dialogues, with content properly annotated by the argumentation schemes used. Note that while [162] increases the amount of content communicated by agents, we aim to make the communications as simple as possible, communicating only the essential information to reach a mutual understanding of that communication.

In [34], the authors aim to connect research made on computational linguistics and argumentation theory, looking for a clear natural language account for argumentation schemes. Thus, they analyse how argumentation schemes fit into categories of the discourse relations in the Peen Discourse Treebank (PDTB).

In [148] the authors propose an argumentation-based approach based on argumentation scheme that can be used in team deliberation dialogues focused on establishing agreements about the best course of action to adopt when considering teamwork. The authors explore the conflicting situation in which agents' actions may be incompatible with existing commitments and norms. One important point in [148] is that agents are able to clarify the nature of the conflicts in a joint plan. The argumentation schemes introduced in that paper are patterns of argument that are commonly used in deliberation dialogue, and they could be incorporated in our framework.

In [131], the authors describe a model for argumentation schemes in agent communication. Their results show advantages of flexibility, scope, knowledge sharing, as well as computational efficiency. Besides the AI literature has interested in non-deductive forms of reasoning, argumentation schemes provide a more fine-grained analysis than other typical approaches within AI. Although argumentation schemes have been investigated by the literature, a more refined representation of such structures of reasoning are necessary to multi-agent systems field, becoming adequate to a computational interpretation and implementation.

The authors [131] point out the following advantages of using argumentation schemes in agent communication:

- The stratification of the belief base. According agents become large (considering agents developed in real-world applications), deduction and search in their data bases (belief bases) becomes computationally expensive. Using argumentation schemes reduces the branching factor, given that a conclusion is reached using only some types of propositions.
- Reduction of the load for the header. Processing an incoming argument is similarly computationally intensive, and this search is reduced given the scheme-based stratification.
- Inter-agent communication. The scheme is judged by the community (society) and arguments instantiated from those have a particular way to be evaluated.
- It allows human-machine communication through the medium of natural language restricted through structural constraints and ontological limits.
- To have agents configure their reasoning capabilities on the bases of the definitions of a set of schemes.

6.2 Enthymemes

In [67], the author presents an argumentation framework for “real arguments”. That work is based on an existing proposal for logic-based argumentation [18], but considering enthymemes, which are arguments with missing premises and seem more like those used by people in general. In the framework for real arguments presented in [67], when an agent wishes a recipient to be aware of an argument (they refer to this argument as *intended argument*), the proponent may send an enthymeme instead of the intended argument. The work describes the processes of encodation (creating an enthymeme from an argument) and decodation (rebuilding the argument from an enthymeme) claiming that by using a ranking information in the so-called cobase (common knowledge) they can obtain a reasonable decoding of an enthymeme (given that an agent could decode an enthymeme using another formula than the original used by the proponent of the arguments). Further, they say that if the recipient and the proponent have identical common knowledge, then the intended argument is one of the decodation [67]. They evaluate enthymemes in terms of quality (premises of the intended argument follow the decodation) and efficiency (premises of the decodation follow the intend argument). Both evaluations allow us to measure the mutation occurring in arguments during the processes of dialogues. However, as mentioned by the authors, to measure mutation in their work it is necessary for the proponent to send the intend argument to the recipient as well (or the recipient needs to send its decodation to the proponent). In our approach for enthymemes, while we improve the computational and linguistic aspect of argumentation-based dialogues using enthymemes, we guarantee that both sides of communication will have the same understanding of what has been communicated, in which enthymeme will be decoded exactly like the intended argument.

In [23], the authors investigate the use of enthymemes, following the work presented in [67], in inquiry dialogues, showing that enthymemes can be managed by the agents involved and how common knowledge can evolve through dialogues. That work seems very interesting, given that the goal of participants in an inquiry dialogue is exactly to share knowledge, in order to jointly construct arguments. In that work, they emphasise that "while humans are constantly handling examples using enthymemes, the logical formalisation that characterises the process remains underdeveloped" [23]. Although the work is concerned with agent-human and human-agent communication through enthymemes, we take some inspiration from [23] in our work. Such work started in [22], but without any reference to enthymemes. The authors state that the decodification is not guaranteed given that agents may have different views on what constitutes common knowledge. The process is only guaranteed when agents have identical beliefs. Our work differs from [23], given that we are interested in agents constructing and exchange enthymemes as a means to make agent communication computationally more efficient, while [23] aimed to allow agents to construct enthymemes together. However, our work takes into consideration some definitions presented in [23], as well as the concepts presented in that work.

In [76], the authors argue that the term "enthymeme" could be misinterpreted from Aristotle. The real meaning of enthymeme could be arguments based on defeasible generalisations. However, considering that the term enthymeme is well established in logic as an argument with missing premises, the authors suggest to maintain that definition, but with the following qualifications: (i) taking into account arguments with missing premises or conclusions that are based on argumentation schemes that are defeasible (instead of only deductive arguments); (ii) considering current work that already treats enthymemes using defeasible argumentation schemes; (iii) keeping in mind Burnyeat's theory that Aristotle was well aware of the importance and uses of defeasible argumentation schemes. Note that our work considers both views of enthymemes, i.e., the implicit information pointed out by the critical questions of an argumentation scheme, as well as explicit premises of arguments omitted when their represent shared knowledge.

In [154], the author says that his three previous papers described below are the three main principles to develop a dialectical theory of enthymemes (including argumentation schemes, arguer's commitments, and common knowledge).

- In [158], the author showed how enthymemes are often based on implicit premises that can be classified as falling under the heading of common knowledge. The work shows several examples of enthymemes that are found in ordinary conversational argumentation. Also, it shows that implicit premises based on common knowledge can be commonly founded in argumentation, although the author did not propose a solution for the inherent problem of enthymemes.
- In [157], the authors showed how argumentation schemes could be applied to an argument found in the text of a discourse. Also, they show how an argumentation scheme can be used to reveal implicit premises needed to make the argument fit the requirements of the scheme. This method of reconstructing enthymemes was shown to be valuable in revealing the needed premises in an argument with implicit premises, even though it was conceded that it did not provide an automated enthymeme system that could be mechanically applied to a given argument in the text of a discourse to reveal any implicit premises or conclusions in the given argument.
- In [155], the author surveyed research on common knowledge in artificial intelligence, showing how argumentation and common knowledge work together.

In [154], Walton presented a model of dialogues, including a common knowledge database with the propositions that no agent would dispute, or have any interest in disputing, given that such information is widely accepted as being true, or at any rate are not widely subject to doubt or disputation. The common knowledge database is described as domain-dependent because what is taken to be common knowledge varies widely depending on the context of the dialogue [154]. According to the authors, the model proposed in [154] has three of the four necessary components. The model has: (i) the set of locution rules, dialogue rules and commitment rules; (ii) a set of propositions the participant agrees to be common knowledge; and (iii) a set of propositions for each participant representing the explicit and implicit commitments. The missing components in the framework, as described by the author [154], are the set of argumentation schemes representing a wide variety of types of arguments used in everyday conversational argumentation, including defeasible schemes as well as deductive and inductive ones. Our work is inspired by Walton's work, but here in the multi-agent system context. Thus, the common knowledge data base Walton refers to in his approach, here is the organisation knowledge, which no agent should dispute. Also, we propose an infrastructure to share argumentation scheme in multi-agent systems, making all agents aware of this set of reasoning patterns, which is a direction pointed out by Walton.

In [77], the author points out that there may be different reasons for an agent not to share some part of its knowledge (i.e., using enthymemes), such as some cost on the communication process. Also, the author says that "if missing formulae to complete the enthymeme are not part of the agent's beliefs, then it will use a badly completed enthymeme in her argumentation framework" [77]. This shows the main problem in using enthymemes, given that the nature of arguments and attacks is not absolute; it depends on the agent's beliefs and on its way to complete enthymemes. The work in [77] is based on the approach to deductive arguments [18, 21] and claims that under the assumption that the agents share all their knowledge and beliefs, the personal beliefs of the agents can be represented as additional arguments and we could obtain a single argumentation framework like in Dung's work [45], representing the whole information about a topic, reaching the acceptable conclusion from that. However, the author thinks that this assumption is too strong for at least three reasons. The author thinks that there may be technical issues with this information sharing, for example, there may be some cost on communication between agents, or the global amount of information in the network may be too important to be stored in a centralized way. Also, for strategical reasons, agents may choose not to share their knowledge and beliefs. In that work, the author argues that the approach in [67] has two main problems: (i) the mapping of common knowledge by an Agent A describes the perception that agent A has of its common knowledge with the other agent. If this perception is wrong, then there could be some exchange of enthymeme that the other agent cannot decode accurately; (ii) even with a good evaluation of the common knowledge by such perception of common knowledge, the choice of a bad threshold could also lead to enthymemes that the other agent cannot decode. They argue that, in realistic situations, agents do not share all their knowledge and beliefs [77]. In our approach for enthymemes, we considered the authors perceptions [67], in which we developed an approach that avoids the problems mentioned in [67].

In [24], the authors claim that whilst humans are constantly handling examples based on enthymemes, proposals for logic-based formalisations of the process remain underdeveloped. Arguing also that deciding how to construct or deconstruct enthymemes is difficult, and proposals for logic-based formalisations of the process remain also underdeveloped. They define properties that characterise aspects of relevance theory (in particular, the idea that an argument is relevant if it maximises cognitive effect whilst minimising cognitive effort) and show that these properties hold for their system of enthymemes. The authors assume that an agent will prefer to use inferences step that require less cognitive effort, so if different conclusions could be taken from the same enthymeme, an agent is able to consider that the others will use the inferences that require less cognitive effort. When an agent receives an enthymeme missing the claim (conclusion), then it looks for the highest ranked parts of its cobase (common knowledge) to try to find a way of meeting the most pressing required information it can, given the premises that have been provided and the shared knowledge. The authors conclude the paper saying that "argumentation is an important cognitive activity that needs to be better understood if we are to build intelligent systems better able to deal with conflicts arising in information and between agents".

In [157], the authors investigate the role of argumentation scheme in enthymemes reconstruction. The study shows that for the ten analysed case studies, they require a less strict standard of reasoning that is defeasible in nature. One possible solution to the enthymeme problem (fill the missing premises) is to make the best interpretation possible (to reconstruct the best/stronger argument possible); however, this could make a (originally) weak argument stronger, which may be distorting it. We took the same directions pointed out in [157], but here, considering a framework for argumentation schemes in multi-agent systems.

6.3 Data-Access Control

In [43], the authors propose a persuasion protocol (also described as an information-seeking protocol by the authors) for access authorisation. In that work, a server and a client argue about seeking and granting authorisation to access some information source. The scenario presented in [43] is based on two agents, a client and a server, where the client tries to access some information held by the server agent and the server agent tries to convince the client agent that it cannot give to it the access. In [43] a ***permission*** is denoted by a function $\texttt{perm}(\texttt{y}, \texttt{x}, \varphi)$, which returns 1 when y can give access to agent x the content of information φ, and 0 otherwise. The link between permissions and arguments is made by defining arguments about to give access to information or not, i.e., defining a tuple $\langle A, y, x, \varphi, i \rangle$ with A the argument for (when $i = 1$) or against (when $i = 0$) a given permission. In addition, the authors, in [43], describe that a ***consistent*** permission should respect the following constraints: (i) y need to has the control of φ; (ii) arguments for and against permission respect the defeat relation; and (iii) the permission is supported by at least one acceptable argument.

An important consideration comparing our work with [43] is that our work does not aim to allow agents to change the permissions. It only allows agents to perceive a wrong classification of the request or an emergency situation in which such access should be granted. In contrast, [43] describes that an information φ has been provided because: (i) the client agent had the permission; or (ii) the client agent did not have the permission, but a persuasion has occurred, all arguments and counter-arguments related to the permission have been exchanged, and the server agent has decided to change the permission the client agent has. We argue that an agent should not be able to change the permissions in a MAS in such easy way.

In [28], the authors investigated argumentation-based reasoning for access control in the context of agents negotiating access to resources or web services in virtual organisations. In particular, the authors describe an approach in which the parties are able to negotiate about which policy to apply, given that it is possible to have more than one way to achieve a security goal. The authors describe that a formal framework for negotiation about data access control needs: (i) a formal representation of the language (arguments about access control); and (ii) a formal representation of the possible moves (request, grant, challenge, defend, etc.) and a interaction protocol to specify the sequences of moves are allowed. Thus, the authors address the formalisation of arguments about interactive and declarative access control situations.

In [42], the authors describe an architecture, communication language and protocol for regulated exchange of information between police officers. Such interactions are negotiation with embedded persuasion dialogues. Considering that in many organisation information has to be exchanged and such activity is normally regulated by law, that approach aims to allow agents interact to promote optimal information exchange while respecting such law. Following [42], the problem of regulated information exchange has two sides (i) one the one hand, the privacy of the persons who are the subjects of the information must be protected; one the other hand, the legitimate interest of the exchanging institutions must be served. Also, while some institutions have as a goal to share and acquire as much information as possible, other institutions have the goal to protect as much information they can in order to protect their own objectives. In the most cases there is a central institution (e.g., the state, the mother company, etc.) which aims to optimal and legitimate information exchange between such institutions, from where several conflicts of interest arise from those diverging goals. Also, the authors describe that, for example, some police departments are reluctant to share investigation information with other departments, even if the sharing of information is allowed. Thus, that work aims to develop automated support for information-exchanging police officers.

In [73], the authors argue that when users' privacy constraints are distributed (most of the cases), the transactions have effects on others and preserving the privacy is more difficult than the ones managed by single users. Thus, they modelled different users from social network as agents, representing their privacy rules, and using argumentation-based reasoning agents to decide whether content should be shared or not. In that work, each agent in the system is equipped with an ontology and the semantic rules that capture its user's privacy constraints, thus when a user decides to put up a content online, the user's agent contacts all those other agents relevant to the content (tagged or mentioned people) to request permission. Each user's agent evaluates the request, and if the content is acceptable, then the content can be shared online. In the case an agent has a concern (the user's privacy constraints are violated), it starts a persuasion dialogue to solve the conflict of opinion. Using an argumentation engine based on assumption-based argumentation (ABA), the authors propose a distributed argumentation for privacy, which they called `PriArg`. The basic idea is that agents can exchange argument until all involved agents decide to stop presenting arguments, and the ABA engine checks if the resulting conclusion is to share or not the post[1]. Other works towards privacy negotiation in social networks are: [86, 140].

[1]We quote the following sentence from that work: "In systems, where transactions are managed by a single user, such as e-commerce systems, preserving privacy of the transactions is merely the capability of access control" [73].

7. CONCLUSION

In this document, we presented a complete (formal and implemented) framework for reasoning and dialogues in multi-agent systems using argumentation schemes. As it can be seen in the relevant literature on argumentation, the studies of argumentation schemes (and enthymemes) in multi-agent systems are still very recent and it is considered as an underdeveloped area. The framework described in this document presents three main contributions to the literature on multi-agent system and argumentation, corresponding to the chapters 3, 4 and 5.

First, in Chapter 3, we introduced a formal consideration of argumentation schemes in multi-agent systems, including: (i) two alternative ways to represent these reasoning patterns into multi-agent systems, extending the organisational specification in such systems with a new dimension for argumentation schemes or sharing argumentation schemes using semantics databases (ontologies), respectively; and (ii) an internal representation for argumentation scheme in Jason agents, which allows them to instantiate arguments from such reasoning patterns using the knowledge available to them. To the best of our knowledge, this work is the first to introduce a formal e general approach for argumentation scheme in multi-agent systems.

Second, in Chapter 4, we extended our previous work on argumentation-based reasoning in order to consider argumentation schemes. That is, we extended the argumentation-based reasoning mechanism developed in [109, 101] in order to consider the internal representation of argumentation schemes in Jason agents. Thus, using the extended argumentation-based reasoning mechanism, agents are able to instantiate arguments from the argumentation schemes and other knowledge available to them (which we called acceptable instances of argumentation schemes). Then, agents are able to check the acceptability of those arguments, considering the constellation of arguments they are able to construct, and the attack relations among such arguments. After, we showed the generality of our framework representing various argumentation schemes from different areas in the literature and implementing agents able to reason about those domains. Further, in order to show how argumentation schemes could be modelled for a particular application domain, we investigated the problem of data access control in multi-agent systems, proposing argumentation schemes that interface agents might use to make decisions about sharing or not information that has been requested. In this exercise, we show a complete process of knowledge engineering towards modelling argumentation schemes for data access control. Furthermore, we evaluated the implementation of the extended argumentation-based reasoning mechanism in order to show how the use of nested argumentation scheme (i.e., argumentation schemes in which premises of arguments are inferred using others argumentation schemes) has influence on the time necessary to agents reach a particular conclusion. This evaluation is very useful to help developers on choices they will make when modelling argumentation schemes in our framework. To the best of our knowledge, the extended argumentation-based reasoning mechanism we developed is the first implemented mechanism that considers a general structure of argumentation schemes in agent-oriented programming languages. That is, agents are able to reason over any argumentation scheme structured in our framework, as well as they are able to reason over any number of argumentation schemes.

Third, in Chapter 5, we described the link between argumentation schemes and dialogues. We started describing the main speech-acts used in the literature of argumentation-based dialogues and our previous work defining the operational semantics for those speech-acts [108, 111]. We also described artifacts we implemented to support argumentation-based dialogues in multi-agent systems. After, we discussed how the particular structure of argumentation schemes allows agents to execute extra moves in argumentation-based dialogues, using the critical questions related to each argumentation scheme. Those extra moves might be considered when defining protocols for argumentation. In order to evaluate our framework in the dialogue perspective, i.e., to evaluate the support that our framework provides when defining and implementing different protocols for argumentation, we have formally defined and implemented two different protocols using our framework. For each protocol, we ran some experiments using different agents' configurations. Our experiments show that our framework is general enough to implement different scenarios of argumentation found in the literature, reaching the same conclusions presented by the authors of those scenarios. Also, considering some characteristics in our framework, we investigated how agents could use more rational communications through using enthymemes, i.e., how agents can rationally omit shared knowledge during communication. Our experiments show that we are able to reduce considerably the amount of content that agents communicate during argumentation-based dialogues using our approach for enthymemes in multi-agent systems. These results are important mainly when considering multi-agent systems that run in distributed environments using restrict network infrastructure. To the best of our knowledge, our work is the first framework to support the implementation of protocols for argumentation, based on argumentation schemes, in multi-agent systems, as well as the first general approach for enthymemes in multi-agent systems.

7.1 Extensions and Future Work

In the previous chapters, we presented only part of the work we have made on argumentation and multi-agent systems. For simplicity and readability, we presented only the core of our framework, and we are going to shortly discuss some of the extensions we have developed in this section. The main extensions focused on (i) using different meta-information available in multi-agent systems to calculate the strength of arguments, defining preferences between argument in argumentation-based reasoning and dialogues; and (ii) the modelling of other agents' mental attitudes in order to take advantage of that information in argumentation-based dialogues;

In a series of papers [82, 83, 85, 84, 107], we have investigated how our approach for argumentation-based reasoning in multi-agent systems, presented in Chapter 4, could be extended in order to consider preferences between arguments. We noted that there are many meta-information produced into multi-agent systems that could be used to calculate a degree of truth to each piece of information that agents acquire during their execution. Thus, we proposed an approach in which agents consider different meta-information to calculate a degree of truth on their beliefs. After, based on this degree of truth on their beliefs, they are able to calculate the strength of each argument they are able to construct using that information. Comparing the strength of arguments, agents are able to define preferences between arguments, deciding conflicts they could not decide without preferences. The main idea in that extended version of our work is that agents will prefer arguments constructed from more updated information, as well as information from more trustworthy sources (agents, sensors in the environments, themselves, etc.). Also, considering agents that define preferences between arguments based on the sources of information used in those arguments, we have proposed a strategy that agents could use during argumentation-based dialogues [112]. In that work, we propose that agents might construct more appropriate arguments to a particular recipient if they consider who are the sources of information the recipient trusts.

Also, in a series of papers [96, 95, 134], we have investigated how agents can model other agents' mental attitudes in order to use this model to take advantage of their opponents during communication. Those studies focused on an interesting topic of research called Theory of Mind, in which an agent is not only able to model other agents' minds but also to simulate other agents' reasoning over that model. Thus, an agent is able to anticipate the consequences of saying something during a dialogue. We have used a simplified version of these studies in [112], in which agents model how much other agents trust each other, in order to construct more appropriate arguments depending on who is the recipient and the recipient's view on trustworthy sources.

Besides the extensions briefly discussed, there are many directions for the research we described in this document. The main directions are listed below:

- **Human-computer Interaction**: arguments from our approach can be easily translated to text and/or voice, requiring only an HCI interface in our framework. This direction for our research might allow us to implement different kinds of applications in which such interaction is necessary, e.g., health care application for people with disability, applications to support learning, Ambient Intelligence, and many others. Also, using enthymemes in human-computer interaction seems a very interesting topic of research, given humans normally communicate enthymemes [156]. It could allow us to implement more rational and efficient communication between humans and computers, in which only the necessary information for a mutual understanding is communicated.

- **Explainable AI**: One of the greatest challenges to the Artificial Intelligence community today is to develop more transparent techniques [133], in which humans are able to understand a decision made by a machine, for example. There are many scenarios in which artificial intelligence will be useful only when transparent methods could be implemented, e.g., support to diagnostics in hospitals, autonomous vehicles, decision-making in critical environments, and many others. As we can easily note, argumentation provides a mechanism that, based on acceptable arguments, allows agents to make decisions, and those decisions can be easily explainable to humans, in which the agent only need to present the arguments used to support that decision. Our framework allows us to implement applications in which agents are able to explain their decision-making and beliefs. In future work, we intend to implement such applications.

- **Ranking Semantics**: the literature of argumentation has discussed the so-called ranking semantics, in which arguments are ranked from the most acceptable to the weakest argument [5]. Our research regarding preferences between arguments seems to be related to ranking semantics. In our future work, we intend to investigate this relation.
- **Theory of Mind and Enthymemes**: a likely direction for our research is combining our approach for theory of mind in multi-agent systems [95, 96, 134] and our approach for enthymemes. Thus, agents could acquire a model of other agents' mental attitudes, e.g., beliefs, and according to this model, they might be able to decide which information they share, omitting such information during communication.
- **Applications and Interfaces**: as described in this document, our main goal was to develop a complete framework for argumentation schemes in multi-agent systems. Using this framework, we are able to implement different applications and interfaces. For example, as described in Chapters 4 and 5, we have modelled argumentation schemes that can be used to implement argumentation-based interface agents for data access control. In our future research, we intend to improve this implementation in order to make it widely applicable, as well as we intend to use our framework to implement applications and interfaces, focusing on the social good.

7.2 Publications

We have published the following papers related to the research presented in this document:

- In [99], we provide an overview of the complete framework for argumentation scheme in multi-agent systems presented in this document.
- In [102], we describe the extended organisational specification presented in section 3.1.
- In [109, 101], we present the development of the argumentation-based reasoning mechanism described in Chapter 4.
- In [108, 110, 111, 106], we present the formal semantics of speech act for argumentation-based dialogues in agent-oriented programming languages, and a brief description of the artifacts implemented to support argumentation based dialogue in multi-agent systems, also presented in Chapter 5.

- In [103, 98], we present our approach for enthymemes in multi-agent systems, also presented in Section 5.6.
- In [100], we present the modelling of argumentation schemes for data access control, also presented in Section 4.5.2.
- In [104, 105], we present an abstract multi-level semantics model used to formalise state transitions in multi-agent systems. This semantics provides an independent view of different levels of abstraction in multi-agent systems, the same idea was used in the formal semantics presented in Section 5.1.
- In [107, 82, 83, 85, 84], we present the extended version of the argumentation-based reasoning mechanism presented in Chapter 4, in order to account preferences between arguments. In [112], we propose strategies agents can use against agents that define preferences between arguments based on the trust relations they have in a multi-agent system.
- In [95, 96, 134], we present an approach for Theory of Mind in agent-oriented programming languages, in which agents are able to model other agents' mental attitudes during communication. This approach will be combined with our work on enthymemes in our future research.
- In [49, 50], we propose an infrastructure to support shared semantics data bases (ontologies) in multi-agent systems, which can be used to shared argumentation schemes as well as other domain-specific knowledge when developing multi-agent system applications.
- In [150], we propose an extended agent architecture for the DBI model, in which agents can execute argumentation-based reasoning using a particular scheme based on the Taulmin's model for argumentation.

REFERENCES

[1] Ali, A.; Fernández, M. "Hybrid enforcement of category-based access control". In: Proceedings of the International Workshop on Security and Trust Management, 2014, pp. 178–182.

[2] Ali, A.; Fernández, M. "A programming language with role-based access control". In: Proceedings of the UG Research in Computer Science - Theory and Applications, 2014, pp. 13-18.

[3] Ali, A.; Fernández, M. "Static enforcement of role-based access control". In: Proceedings of the International Workshop on Automated Specification and Verification of Web Systems, 2014, pp. 36–50.

[4] Alves, S.; Fernández, M. "A framework for the analysis of access control policies with emergency management". *Electronic Notes in Theoretical Computer Science*, vol. 312, Apr 2015, pp. 89–105.

[5] Amgoud, L.; Ben-Naim, J. "Ranking-based semantics for argumentation frameworks". In: Proceedings of the International Conference on Scalable Uncertainty Management, 2013, pp. 134–147.

[6] Amgoud, L.; Besnard, P. "Bridging the gap between abstract argumentation systems and logic". In: Proceedings of the International Conference on Scalable Uncertainty Management, 2009, pp. 12–27.

[7] Amgoud, L.; Maudet, N.; Parsons, S. "Modeling dialogues using argumentation". In: Proceedings of the International Conference on MultiAgent Systems, 2000, pp. 31–38.

[8] Amgoud, L.; Parsons, S.; Maudet, N. "Arguments, dialogue, and negotiation". In: Proceedings of the European Conference on Artificial Intelligence, 2000, pp. 338–342.

[9] Atkinson, K.; Bench-Capon, T.; McBurney, P. "Justifying practical reasoning". In: Proceedings of the International Workshop on Computational Models of Natural Argument, 2004, pp. 87–90.

[10] Atkinson, K.; Bench-Capon, T.; McBurney, P. "Computational representation of practical argument". *Synthese*, vol. 152–2, Sep 2006, pp. 157–206.

[11] Barker, S. "The next 700 access control models or a unifying meta-model?" In: Proceedings of the ACM Symposium on Access Control Models and Technologies, 2009, pp. 187–196.

[12] Bedi, P.; Vashisth, P. "Extending speech-act based communication to enable argumentation in cognitive agents". In: Advances in Computing, Communication and Control, Springer, 2011, pp. 25–40.

[13] Bellifemine, F.; Bergenti, F.; Caire, G.; Poggi, A. "Jade—-a java agent development framework". In: Multi-Agent Programming, Springer, 2005, pp. 125–147.

[14] Bench-Capon, T. J. "Specification and implementation of toulmin dialogue game". In: Proceedings of International Conference on Legal Knowledge and Information Systems, 1998, pp. 5–20.

[15] Bench-Capon, T. J. M. "Value based argumentation frameworks". *CoRR*, vol. cs.AI/0207059, Jul 2002, pp. 1–10.

[16] Bentahar, J.; Alam, R.; Maamar, Z. "An argumentation-based protocol for conflict resolution". In: Proceedings of the Workshop on Knowledge Representation for Agents and MultiAgent Systems, 2008, pp. 19–35.

[17] Berariu, T. "An argumentation framework for bdi agents". In: Studies in Computational Intelligence, Springer, 2014, pp. 343–354.

[18] Besnard, P.; Hunter, A. "A logic-based theory of deductive arguments", *Artificial Intelligence.* vol. 128-1, May 2001, pp. 203–235.

[19] Besnard, P.; Hunter, A. "Elements of Argumentation". The MIT Press, 2008, 298p.

[20] Besnard, P.; Hunter, A. "Argumentation based on classical logic". In: Argumentation in Artificial Intelligence, Springer, 2009, pp. 133–152.

[21] Besnard, P.; Hunter, A. "Constructing argument graphs with deductive arguments: a tutorial". *Argument & Computation*, vol. 5-1, Jan 2014, pp 5–30.

[22] Black, E.; Hunter, A. "A generative inquiry dialogue system". In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2007, pp. 1010–1017.

[23] Black, E.; Hunter, A. "Using enthymemes in an inquiry dialogue system". In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Dystems, 2008, pp. 437–444.

[24] Black, E.; Hunter, A. "A relevance-theoretic framework for constructing and deconstructing enthymemes". *Journal of Logic and Computation*, vol. 22-1, Feb 2009, pp. 55–78.

[25] Blanger, L.; Junior, V.; Danner, V.; Panisson, A. R. "A taxi service approach using multi-agent systems". In: Proceedings of the II Congresso Internacional de Gestão, Tecnologia e Inovação, 2016, pp. 1–15.

[26] Blanger, L.; Junior, V.; Jevinski, C. J.; Panisson, A. R.; Bordini, R. H. "Improving the performance of taxi service applications using multi-agent systems techniques". In: Proceedings of the Encontro Nacional de Inteligência Artificial e Computacional, 2017, pp. 217–229.

[27] Blanger, L.; Junior, V.; Panisson, A. R. "Uma aplicação para gerenciamento de motoristas autônomos: Usufruindo da escalabilidade oferecida por sistemas multiagentes". In: Proceedings of the Computer on the Beach, 2017, pp. 80–89.

[28] Boella, G.; Hulstijn, J.; Van Der Torre, L. "Argumentation for access control". In: Proceedings of the Congress of the Italian Association for Artificial Intelligence, 2005, pp. 86–97.

[29] Boella, G.; van der Torre, L. "Permissions and obligations in hierarchical normative systems". In: Proceedings of the International Conference on Artificial Intelligence and Law, 2003, pp. 109–118.

[30] Boissier, O.; Balbo, F.; Badeig, F. "Controlling multi-party interaction within normative multi-agent organizations". In: Coordination, Organizations, Institutions, and Norms in Agent Systems VI, Springer, 2011, pp. 357–376.

[31] Boissier, O.; Bordini, R. H.; Hübner, J. F.; Ricci, A.; Santi, A. "Multi-agent oriented programming with jacamo". *Science of Computer Programming*, vol. 78–6, Jun 2013, pp. 747–761.

[32] Bordini, R. H.; Dastani, M.; Dix, J.; Seghrouchni, A. E. F. "Multi-Agent Programming: Languages, Tools and Applications". Springer, 2009, 389p.

[33] Bordini, R. H.; Hübner, J. F.; Wooldridge, M. "Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)". John Wiley & Sons, 2007, 292p.

[34] Cabrio, E.; Tonelli, S.; Villata, S. "A natural language account for argumentation schemes". In: Proceedings of the Congress of the Italian Association for Artificial Intelligence, 2013, pp. 181–192.

[35] Caminada, M.; Amgoud, L. "On the evaluation of argumentation formalisms". *Artificial Intelligence*, vol. 171–5, Apr 2007, pp. 286–310.

[36] Chesñevar, C.; Modgil, S.; Rahwan, I.; Reed, C.; Simari, G.; South, M.; Vreeswijk, G.; Willmott, S. "Towards an argument interchange format". *The Knowledge Engineering Review*, vol. 21–04, Dec 2006, pp. 293–316.

[37] Coetzee, L.; Eksteen, J. "The internet of things promise for the future? an introduction". In: Proceedings of the IST-Africa Conference, 2011, pp. 1–9.

[38] Costa, A. C. d. R.; Dimuro, G. P. "Introducing social groups and group exchanges in the poporg model". In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 1297–1298.

[39] da Rocha Costa, A. C.; Dimuro, G. P. "A minimal dynamical mas organization model". In: Handbook of Research on Multia-Agent Systems: Semantics and Dynamics of Organizational Models, 2009, pp. 419–445.

[40] de Matos, E.; Amaral, L. A.; Tiburski, R. T.; Schenfeld, M. C.; de Azevedo, D. F.; Hessel, F. "A sensing-as-a-service context-aware system for internet of things environments". In: Proceedings of the IEEE Annual Consumer Communications & Networking Conference, 2017, pp. 724–727.

[41] Dignum, F.; Dunin-Keplicz, B.; Verbrugge, R. "Creating collective intention through dialogue". *Logic Journal of IGPL*. vol. 9–2, Mar 2001, pp. 289–304.

[42] Dijkstra, P.; Bex, F.; Prakken, H.; de Vey Mestdagh, K. "Towards a multi-agent system for regulated information exchange in crime investigations". *Artificial Intelligence and Law*, vol. 13–1, Mar 2005, pp. 133–151.

[43] Doutre, S.; McBurney, P.; Perrussel, L.; Thévenin, J.-M. "Arguing for gaining access to information". In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2007, pp. 20–23.

[44] Dung, P. M. "An argumentation-theoretic foundation for logic programming". *The Journal of logic programming*, vol. 22–2, Feb 1995, pp. 151–177.

[45] Dung, P. M. "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games". *Artificial Intelligence*, vol. 77, Sep 1995, pp. 321–357.

[46] Dung, P. M.; Kowalski, R. A.; Toni, F. "Assumption-based argumentation". In: Argumentation in Artificial Intelligence, Springer, 2009, pp. 199–218.

[47] Fagundes, M. S.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Interaction patterns in a multi-agent organisation to support shared tasks". In: Proceedings of the International and Interdisciplinary Conference on Modeling and Using Context, 2013, pp. 364–370.

[48] Ferraiolo, D. F.; Sandhu, R.; Gavrila, S.; Kuhn, D. R.; Chandramouli, R. "Proposed nist standard for role-based access control". *ACM Transactions on Information and System Security*, vol. 4–3, Aug 2001, pp. 224–274.

[49] Freitas, A.; Panisson, A. R.; Hilgert, L.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Integrating ontologies with multi-agent systems through CArtAgO artifacts". In: Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2015, pp. 143–150.

[50] Freitas, A.; Panisson, A. R.; Hilgert, L.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Applying ontologies to the development and execution of multi-agent systems". *Web Intelligence*, vol. 15-4, Jan 2017, pp. 291–302.

[51] Gabbay, D. M. “Labelled deductive systems”. Oxford University Press, 1996, 512p.

[52] García, A. J.; Rotstein, N. D.; Tucat, M.; Simari, G. R. “An argumentative reasoning service for deliberative agents”. In: Proceedings of the International Conference on Knowledge Science, Engineering and Management, 2007, pp. 128–139.

[53] García, A. J.; Simari, G. R. “Defeasible logic programming: An argumentative approach”. *Theory and practice of logic programming*, vol. 4–1, Feb 2004, pp. 95–138.

[54] García, A. J.; Simari, G. R. “Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers”. *Argument & Computation*, vol. 5–1, Jan 2014, pp. 63–88.

[55] Gordon, T. F.; Walton, D. “Formalizing balancing arguments”. In: Proceedings of the International Conference on Computational Models of Arguments, 2016, pp. 203–222.

[56] Gorodetsky, V. “Internet of agents: From set of autonomous agents to network object”. In: Proceedings of the International Workshop on the Internet of agents, 2017, pp. 1–10.

[57] Governatori, G.; Maher, M. J.; Antoniou, G.; Billington, D. “Argumentation semantics for defeasible logic”. *Journal of Logic and Computation*, vol 14–5, Oct 2004, pp. 675–702.

[58] Green, N. “Implementing argumentation schemes as logic programs”. In: Proceedings of the International Workshop on Computational Models of Natural Argument, 2016, pp. 1–7

[59] Herrlich, H.; Strecker, G. “Category theory”. Heldermann Verlag, 1973, 402p.

[60] Hitchcock, D. “Does the traditional treatment of enthymemes rest on a mistake?”, In: On Reasoning and Argument, Springer, 2017, pp. 57–79.

[61] Horridge, M.; Bechhofer, S. “The OWL API: A Java API for OWL ontologies”. *Semantic Web*, vol. 2–1, Jan 2011, pp. 11–21.

[62] Hu, V. C.; Kuhn, D. R.; Ferraiolo, D. F. “Attribute-based access control”. *Computer*, vol. 48–2, Jan 2015, pp. 85–88.

[63] Hübner, A.; Dimuro, G. P.; da Rocha Costa, A. C.; Mattos, V. “A dialogic dimension for the moise+ organizational model”. In: Proceedings of the Southern Conference on Computational Modeling, 2010, pp. 1–15.

[64] Hübner, J. F.; Boissier, O.; Bordini, R. H. “A normative programming language for multi-agent organisations”. *Annals of Mathematics and Artificial Intelligence*, vol 62–1-2, Jun 2011, pp. 27–53.

[65] Hübner, J. F.; Boissier, O.; Kitio, R.; Ricci, A. “Instrumenting multi-agent organisations with organisational artifacts and agents”. *Autonomous Agents and Multi-Agent Systems*, vol. 20–3, May 2010, pp. 369–400.

[66] Hubner, J. F.; Sichman, J. S.; Boissier, O. “Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels”. *International Journal of Agent-Oriented Software Engineering*, vol. 1–3, Dec 2007, pp. 370–395.

[67] Hunter, A. “Real arguments are approximate arguments”. In: Proceedings of the Association for the Advancement of Artificial Intelligence, 2007, pp. 66–71.

[68] Hussain, A.; Toni, F. “On the benefits of argumentation for negotiation-preliminary version”. In: Proceedings of European Workshop on Multi-Agent Systems, 2008, pp. 1–20.

[69] Karafili, E.; Kakas, A. C.; Spanoudakis, N. I.; Lupu, E. C. “Argumentation-based security for social good”. In: Proceedings of the Association for the Advancement of Artificial Intelligence Fall Symposium Series, 2017, pp. 1–7.

[70] Karafili, E.; Lupu, E. C.; Cullen, A.; Williams, B.; Arunkumar, S.; Calo, S. “Improving data sharing in data rich environments”. In: Proceedings of the IEEE International Conference on Big Data, 2017, pp. 2998–3005.

[71] Karafili, E.; Spanaki, K.; Lupu, E. C. “An argumentation reasoning approach for data processing”. *Computers in Industry*, vol. 94, Jan 2018, pp. 52–61.

[72] Kökciyan, N.; Sassoon, I.; Young, A. P.; Chapman, M.; Porat, T.; Ashworth, M.; Curcin, V.; Modgil, S.; Parsons, S.; Sklar, E. “Towards an argumentation system for supporting patients in self-managing their chronic conditions”. In: Proceedings of the Association for the Advancement of Artificial Intelligence Joint Workshop on Health Intelligence, 2018, pp. 1–8.

[73] Kökciyan, N.; Yaglikci, N.; Yolum, P. “An argumentation approach for resolving privacy disputes in online social networks”. *ACM Transactions on Internet Technology*, vol. 17–3, Jul 2017, pp. 27–49.

[74] Koster, A.; Bazzan, A. L.; de Souza, M. “Liar liar, pants on fire; or how to use subjective logic and argumentation to evaluate information from untrustworthy sources”. *Artificial Intelligence Review*, vol. 48–2, Aug 2016, pp. 1–17.

[75] Li, X.; Xuan, Z.; Wen, L. “Research on the architecture of trusted security system based on the internet of things”. In: Proceedings of the Intelligent Computation Technology and Automation, 2011, pp. 1172–1175.

[76] Macagno, F.; Walton, D. “Enthymemes, argumentation schemes, and topics”. *Logique et Analyse*, vol. 205, Mar 2009, pp. 39–56.

[77] Mailly, J.-G. “Using enthymemes to fill the gap between logical argumentation and revision of abstract argumentation frameworks”. *CoRR*, vol. abs/1603.08789-1, Mar 2016, pp. 1–11.

[78] Maudet, N.; Parsons, S.; Rahwan, I. “Argumentation in multi-agent systems: Context and recent developments”. In: Proceedings of the Workshop on Argumentation in Multi-Agent Systems, 2006, pp. 1–16.

[79] McBurney, P.; Parsons, S. “Games that agents play: A formal framework for dialogues between autonomous agents”. *Journal of Logic, Language and Information*, vol. 11, Jun 2001, pp. 315–334.

[80] McBurney, P.; Parsons, S. “Dialogue games in multi-agent systems”. *Informal Logic*, vol. 22, May 2002, pp. 10-45.

[81] McBurney, P.; Parsons, S. “Locutions for argumentation in agent interaction protocols”. In: Proceedings of the International Workshop on Agent Communication, 2004, pp. 209–225.

[82] Melo, V. S.; Panisson, A. R.; Bordini, R. H. “Argumentation-based reasoning using preferences over sources of information”. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, 2016, pp. 1337–1338.

[83] Melo, V. S.; Panisson, A. R.; Bordini, R. H. “Trust on beliefs: Source, time and expertise”. In: Proceedings of the International Workshop on Trust in Agent societies, 2016, pp. 1–12.

[84] Melo, V. S.; Panisson, A. R.; Bordini, R. H. “Meta-information and argumentation in multi-agent systems”. *iSys-Revista Brasileira de Sistemas de Informação*, vol 10–3, Sep 2017, pp. 74–97.

[85] Melo, V. S.; Panisson, A. R.; Bordini, R. H. “Mirs: A modular approach for using meta-information in agent-oriented programming languages”. In: Proceedings of the International Workshop on Trust in Agent societies, 2017, pp. 1–15.

[86] Mester, Y.; Kökciyan, N.; Yolum, P. “Negotiating privacy constraints in online social networks”. In: Advances in Social Computing and Multiagent Systems, Springer, 2015, pp. 112–129.

[87] Modgil, S.; Prakken, H. “A general account of argumentation with preferences”. *Artificial Intelligence*, vol. 195, Feb 2013, pp. 361–397.

[88] Modgil, S.; Prakken, H. “The aspic+ framework for structured argumentation: a tutorial”. *Argument & Computation*, vol. 5–1, Jan 2014, pp. 31–62.

[89] Modgil, S.; Tolchinsky, P.; Cortés, U. “Towards formalising agent argumentation over the viability of human organs for transplantation”. In: Proceedings of the Mexican International Conference on Artificial Intelligence, 2005, pp. 928–938.

[90] Nettel, A. “The enthymeme between persuasion and argumentation”. In: Proceedings of the Conference on Argumentation of the International Society for the Study of Argumentation, 2011, pp. 1359–1365.

[91] Nute, D. “Defeasible Prolog”. In: Artificial Intelligence Programs, University of Georgia, 1993, pp 105-112.

[92] Nute, D. “Defeasible logic.”. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press, 1994, pp. 353–395.

[93] Nute, D. “Defeasible logic”. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press, 2001, pp. 353–395.

[94] Oliva, E.; Viroli, M.; Omicini, A.; McBurney, P. “Argumentation and artifact for dialogue support”. In: Proceedings of the International Workshop on Argumentation in Multi-Agent Systems, 2009, pp. 107–121.

[95] Panisson, A.; Sarkadi, S.; McBurney, P.; Parsons, S.; Bordini, R. “Lies, Bullshit, and Deception in Agent-Oriented Programming Languages”. CEUR-WS, vol. 2154, Jun 2018, pp. 50–61.

[96] Panisson, A.; Sarkadi, S.; McBurney, P.; Parsons, S.; Bordini, R. “On the Formal Semantics of Theory of Mind in Agent Communication”. In: Proceedings of the International Conference on Agreement Technologies, 2018, pp 1–15.

[97] Panisson, A. R. “Argumentation-based dialogues for task reallocation among rational agents”. Master’s Thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2015, 95p.

[98] Panisson, A. R. “Argumentation schemes and enthymemes in multi-agent systems”. In: Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 1849–1850.

[99] Panisson, A. R. “Towards a framework for argumentation schemes in multi-agent systems”. In: Proceedings of the Summer School on Argumentation: Computational and Linguistic Perspectives, 2018, pp. 21–23.

[100] Panisson, A. R.; Ali, A.; McBurney, P.; Bordini, R. H. “Argumentation schemes for data access control”. In: Proceedings of the International Conference on Computational Models of Argument, 2018, pp 1–8.

[101] Panisson, A. R.; Bordini, R. H. “Knowledge representation for argumentation in agent-oriented programming languages”. In: Proceedings of the Brazilian Conference on Intelligent Systems, 2016, pp. 1–6.

[102] Panisson, A. R.; Bordini, R. H. “Argumentation schemes in multi-agent systems: A social perspective”. In: Proceedings of the International Workshop on Engineering Multi-Agent Systems, 2017, pp. 92–108.

[103] Panisson, A. R.; Bordini, R. H. “Uttering only what is needed: Enthymemes in multi-agent systems”. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, 2017, pp. 1670–1672.

[104] Panisson, A. R.; Bordini, R. H.; da Rocha Costa, A. C. "Towards multi-level semantics for multi-agent systems". In: Proceedings of the Workshop-Escola de Informática Teórica, 2015, pp. 230–237.

[105] Panisson, A. R.; Bordini, R. H.; da Rocha Costa, A. C. "Multi-level semantics with vertical integrity constraints". In: Proceedings of the European Conference on Artificial Intelligence, 2016, pp. 1708–1709.

[106] Panisson, A. R.; Freitas, A.; Schmidt, D.; Hilgert, L.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Arguing About Task Reallocation Using Ontological Information in Multi-Agent Systems". In: Proceedings of the International Workshop on Argumentation in Multiagent Systems, 2015, pp. 1–15.

[107] Panisson, A. R.; Melo, V. S.; Bordini, R. H. "Using preferences over sources of information inargumentation-based reasoning". In: Proceedings of the Brazilian Conference on Intelligent Systems, 2016, pp. 7–13.

[108] Panisson, A. R.; Meneguzzi, F.; Fagundes, M. S.; Vieira, R.; Bordini, R. "Formal semantics of speech acts for argumentative dialogues". In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2014, pp. 1437–1438.

[109] Panisson, A. R.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "An Approach for Argumentation-based Reasoning Using Defeasible Logic in Multi-Agent Programming Languages". In: Proceedings of the International Workshop on Argumentation in Multiagent Systems, 2014, pp. 1–15.

[110] Panisson, A. R.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Towards practical argumentation-based dialogues in multi-agent systems". In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2015, pp. 151–158.

[111] Panisson, A. R.; Meneguzzi, F.; Vieira, R.; Bordini, R. H. "Towards practical argumentation in multi-agent systems". In: Proceedings of the Brazilian Conference on Intelligent Systems, 2015, pp. 98–103.

[112] Panisson, A. R.; Parsons, S.; McBurney, P.; Bordini, R. H. "Choosing appropriate arguments from trustworthy sources". In: Proceedings of the International Conference on Computational Models of Argument, 2018, pp 9–17.

[113] Parsons, S.; Atkinsonb, K.; Haighc, K.; Levittd, K.; Rowed, P. M. J.; Singhf, M. P.; Sklara, E. "Argument schemes for reasoning about trust". In: Proceedings of the International Conference on Computational Models of Argument, 2012, pp. 20–35.

[114] Parsons, S.; McBurney, P. "Argumentation-based dialogues for agent co-ordination". *Group Decision and Negotiation*, vol. 12–5, Sep 2003, pp. 415–439.

[115] Parsons, S.; McBurney, P.; Sklar, E.; Wooldridge, M. “On the relevance of utterances in formal inter-agent dialogues”. In: Proceedings of the International Workshop on Argumentation in Multi-Agent Systems, 2007, pp. 47–62.

[116] Parsons, S.; Wooldridge, M.; Amgoud, L. “An analysis of formal inter-agent dialogues”. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2002, pp. 394–401.

[117] Parsons, S.; Wooldridge, M.; Amgoud, L. “On the outcomes of formal inter-agent dialogues”. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2003, pp. 616–623.

[118] Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. “Context aware computing for the internet of things: A survey”. *IEEE Communications Surveys & Tutorials*, vol. 16–1, Jan 2014, pp. 414–454.

[119] Perrussel, L.; Doutre, S.; Thévenin, J.-M.; McBurney, P. “A persuasion dialog for gaining access to information”. In: Proceedings of the International Workshop on Argumentation in Multi-Agent Systems, 2007, pp. 63–79.

[120] Plotkin, G. D. “A structural approach to operational semantics”. University of Aarhus, 1981, 133p.

[121] Pollock, J. L. “Defeasible reasoning”. *Cognitive science*, vol. 11–4, Oct 1987, pp. 481–518.

[122] Pollock, J. L. “Cognitive carpentry: A blueprint for how to build a person”. Mit Press, 1995, 391p.

[123] Prakken, H. “An abstract framework for argumentation with structured arguments”. *Argument and Computation*, vol. 1–2, Jun 2011, pp. 93–124.

[124] Prakken, H.; Vreeswijk, G. “Logics for defeasible argumentation”. In: Handbook of Philosophical Logic, Dordrecht etc., 2002, pp. 219–318.

[125] Rahwan, I.; Amgoud, L. “An argumentation based approach for practical reasoning”. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 347–354.

[126] Rahwan, I.; Ramchurn, S. D.; Jennings, N. R.; Mcburney, P.; Parsons, S.; Sonenberg, L. “Argumentation-based negotiation”. *The Knowledge Engineering Review*, vol. 18–4, Dec 2004, pp. 343–375.

[127] Rahwan, I.; Simari, G. R.; van Benthem, J. “Argumentation in artificial intelligence”. Springer, 2009, vol. 47, 493p.

[128] Rao, A. S. “AgentSpeak(L): BDI agents speak out in a logical computable language”. In: Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi-agent Wrld: Agents Breaking Away: Agents Breaking Away, 1996, pp. 42–55.

[129] Reed, C.; Rowe, G. “Araucaria: Software for Puzzles in Argument Diagramming and XML”. (Technical Report), Department of Applied Computing. University of Dundee, 2001, 21p.

[130] Reed, C.; Rowe, G. “Araucaria: Software for argument analysis, diagramming and representation”. *International Journal on Artificial Intelligence Tools*, vol 13–04, Dec 2004, pp. 961–979.

[131] Reed, C.; Walton, D. “Towards a formal and implemented model of argumentation schemes in agent communication”. *Autonomous Agents and Multi-Agent Systems*, vol. 11–2, Sep 2005, pp. 173–188.

[132] Ricci, A.; Piunti, M.; Viroli, M. “Environment programming in multi-agent systems: An artifact-based perspective”. *Autonomous Agents and Multi-Agent Systems*, vol. 23–2, Sep 2011, pp. 158–192.

[133] Sample, I. “Computer says no: why making ais fair, accountable and transparent is crucial”. *The Guardian*, Vol. 5, Nov 2017, pp. 1–15.

[134] Sarkadi, S.; Panisson, A.; Bordini, R.; McBurney, P.; Parsons, S. “Towards an Approach for Modelling Uncertain Theory of Mind in Multi-Agent Systems”. In: Proceedings of the International Conference on Agreement Technologies, 2018, pp. 16-31.

[135] Sarma, A. C.; Girão, J. “Identities in the future internet of things”. *Wireless personal communications*, vol. 49–3, May 2009, pp. 353–363.

[136] Schmidt, D.; Panisson, A. R.; Freitas, A.; Bordini, R. H.; Meneguzzi, F.; Vieira, R. “An ontology-based mobile application for task managing in collaborative groups”. In: Proceedings of the Florida Artificial Intelligence Research Society Conference, 2016, pp. 522–526.

[137] Searle, J. R. “Speech Acts: An Essay in the Philosophy of Language”. Cambridge University Press, 1969, 214p.

[138] Sivaraman, V.; Gharakheili, H. H.; Vishwanath, A.; Boreli, R.; Mehani, O. “Network-level security and privacy control for smart-home iot devices”. In: Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2015, pp. 163–167.

[139] Sklar, E. I.; Azhar, M. Q. “Argumentation-based dialogue games for shared control in human-robot systems”. *Journal of Human-Robot Interaction*, vol. 4–3, Dec 2015, pp. 120–148.

[140] Such, J. M.; Rovatsos, M. “Privacy policy negotiation in social media”. *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11-1, Apr 2016, pp. 4–20.

[141] Sundmaeker, H.; Guillemin, P.; Friess, P.; Woelfflé, S. “Vision and challenges for realising the internet of things”. In: Cluster of European Research Projects on the Internet of Things, European Commision, vol. 3-3, 2010, pp. 34–36.

[142] Tolchinsky, P.; Atkinson, K.; McBurney, P.; Modgil, S.; Cortés, U. “Agents deliberating over action proposals using the proclaim model”. In: Proceedings of the International Central and Eastern European Conference on Multi-Agent Systems, 2007, pp. 32–41.

[143] Tolchinsky, P.; Cortés, U.; Nieves, J.; López-Navidad, A.; Caballero, F. “Using arguing agents to increase the human organ pool for transplantation”. In: Proceedings of the Workshop on Agents Applied in Health Care, 2005, pp. 1–11.

[144] Toni, F. “A tutorial on assumption-based argumentation”. *Argument & Computation*, vol. 5-1, Jan 2014, pp. 89–117.

[145] Toniolo, A.; Cerutti, F.; Oren, N.; Norman, T. J.; Sycara, K. “Making informed decisions with provenance and argumentation schemes”. In: Proceedings of the International Workshop on Argumentation in Multi-Agent Systems, 2014, pp. 1–20.

[146] Toniolo, A.; Norman, T. J.; Sycara, K. “On the benefits of argumentation schemes in deliberative dialogue”. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, 2012, pp. 1409–1410.

[147] Toniolo, A.; Norman, T. J.; Sycara, K. P. “Argumentation schemes for collaborative planning”. In: Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems, 2011, pp. 323–335.

[148] Toniolo, A.; Norman, T. J.; Sycara, K. P. “Argumentation schemes for policy-driven planning”. In: Proceedings of the International Workshop on the Theory and Applications of Formal Argumentation, 2011, pp. 1–12.

[149] Toulmin, S. E. “The uses of argument”. Cambridge University Press, vol. 34, 245p.

[150] V. O. Gabriel, D. F. Adamatti, A. R. P. C. Z. B. R. H. B. “Argumentation-based reasoning in BDI agents using toulmin’s model”. In: Proceedings of the Brazilian Conference on Intelligent Systems, 2018, pp. 42–48.

[151] Vieira, R.; Moreira, Á.; Wooldridge, M.; Bordini, R. H. “On the formal semantics of speech-act based communication in an agent-oriented programming language”. *Journal of Artificial Intelligence Reearch*,vol. 29-1, Jun 2007, pp. 221–267.

[152] Vreeswijk, G. A. “Abstract argumentation systems”. *Artificial Intelligence*, vol. 90-1-2, Feb 1997, pp. 225–279.

[153] Walton, D. "Argumentation schemes for presumptive reasoning". Routledge, 1996, 232p.

[154] Walton, D. "The three bases for the enthymeme: A dialogical theory". *Journal of Applied Logic*, vol. 6-3, Sep 2008, pp. 361-379.

[155] Walton, D.; Macagno, F. "Common knowledge in argumentation". In: Studies in Communication Sciences, vol. 6-1, 2006, pp. 3-26.

[156] Walton, D.; Reed, C.; Macagno, F. "Argumentation Schemes". Cambridge University Press, 2008, 456p.

[157] Walton, D.; Reed, C. A. "Argumentation schemes and enthymemes". *Synthese*, vol. 145-3, Jul 2005, pp. 339-370.

[158] Walton, D. N. "Enthymemes, common knowledge, and plausible inference". *Philosophy and rhetoric*, vol. 34-2, Jan 2001, pp. 93-112.

[159] Wanderley, G. M. P.; Abel, M.-H.; Barthès, J.-P.; Paraiso, E. C. "A core architecture for developing systems of systems". In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2017, pp 1-10.

[160] Wang, K.; Bao, J.; Wu, M.; Lu, W. "Research on security management for internet of things". In: Proceedings of the International Conference on Computer Application and System Modeling, 2010, pp 115-133.

[161] Weber, R. H. "Internet of things-new security and privacy challenges". *Computer law & security review*, vol. 26-1, Jan 2010, pp. 23-30.

[162] Wells, S. "Supporting argumentation schemes in argumentative dialogue games". *Studies in Logic, Grammar and Rhetoric*, vol. 36-1, Mar 2014, pp. 171-191.

[163] Wells, S.; Reed, C. A. "A domain specific language for describing diverse systems of dialogue". *Journal of Applied Logic*, vol. 10-4, Dec 2012, pp. 309-329.

[164] Wooldridge, M. "An introduction to multiagent systems". John Wiley & Sons, 2009, 461p.

[165] Wyner, A.; Atkinson, K.; Bench-Capon, T. "A functional perspective on argumentation schemes". In: Proceedings of the International Workshop on Argumentation in Multi-Agent Systems, 2012, pp. 203-222.

[166] Zaslavsky, A.; Perera, C.; Georgakopoulos, D. "Sensing as a service and big data". *CoRR*, vol. abs/1301.0159, Jan 2013, pp. 1-8.

[167] Zatelli, M. R.; Hübner, J. F. "The interaction as an integration component for the jacamo platform". In: Proceedings of the International Workshop on Engineering Multi-Agent Systems, 2014, pp. 431-450.

[168] Zhao, K.; Ge, L. "A survey on the internet of things security". In: Proceedings of the International Conference on Computational Intelligence and Security, 2013, pp. 663–667.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br