Argument & Computation 12 (2021) 357–395
DOI 10.3233/AAC-210555
IOS Press

A computational model of argumentation schemes for multi-agent systems

Alison R. Panisson [a,*], Peter McBurney [b] and Rafael H. Bordini [c]
[a] *Department of Computing, UFSC, Araranguá, Brazil*
E-mail: alison.panisson@ufsc.br
[b] *Department of Informatics, KCL, London, UK*
E-mail: peter.mcburney@kcl.ac.uk
[c] *School of Technology, PUCRS, Porto Alegre, Brazil*
E-mail: rafael.bordini@pucrs.br

Abstract. There are many benefits of using argumentation-based techniques in multi-agent systems, as clearly shown in the literature. Such benefits come not only from the expressiveness that argumentation-based techniques bring to agent communication but also from the reasoning and decision-making capabilities under conditions of conflicting and uncertain information that argumentation enables for autonomous agents. When developing multi-agent applications in which argumentation will be used to improve agent communication and reasoning, argumentation schemes (reasoning patterns for argumentation) are useful in addressing the requirements of the application domain in regards to argumentation (e.g., defining the scope in which argumentation will be used by agents in that particular application). In this work, we propose an argumentation framework that takes into account the particular structure of argumentation schemes at its core. This paper formally defines such a framework and experimentally evaluates its implementation for both argumentation-based reasoning and dialogues.

Keywords: Argumentation, multi-agent systems, argumentation schemes, agent-oriented programming languages

1. Introduction

Argumentation schemes are patterns for arguments (or for inferences), representing the structure of common types of arguments used both in everyday discourse as well as in special contexts such as legal and scientific reasoning [89]. Argumentation schemes have become a well-known concept from Douglas Walton's book "*Argumentation schemes for presumptive reasoning*" [87], a seminal contribution to the literature on argumentation, in which Walton argues about common forms of arguments, considered "valid" or correct, addressing the question of what are those forms of arguments or so-called argumentation schemes. Walton's work on argumentation schemes became influential across a range of areas of study, e.g., legal and scientific argumentation [89], as well as being a central element in the development of argumentation in computer science, in particular artificial intelligence [41,67].

Considering that argumentation is a complex and important ability that demonstrates cognitive capacity [5], the Artificial Intelligence (AI) community sees argumentation schemes as an important component to model and implement intelligent-agent technologies embedding such cognitive capacity, enabling

[*]Corresponding author. E-mail: alison.panisson@ufsc.br.

argument mining [28,35,36], agent decision-making, reasoning, and communication [32,33,77,82]. Applying such technologies on a variety of domains such as legal reasoning, medical argumentation, e-government, debating technologies, and many others [5,69,86].

Particularly in Multi-Agent Systems (MAS), argumentation plays an important role in both agent reasoning and communication [37], providing an expressive communication method [33], as well as a powerful reasoning mechanism under conditions of conflicting and uncertain information [5]. Consequently, interesting multi-agent system applications using argumentation-based techniques have been developed in the last few years, for example [32,46,52,78]. Most of those applications define argumentation schemes (often only one) that are used by agents to instantiate arguments used for reasoning and/or communication, fulfilling the application needs regarding argumentation. However, the link between argumentation schemes and the (few) practical approaches and frameworks for argumentation, particularly in multi-agent systems, has not been fully investigated in the literature. We believe that it is necessary to address the particular structure of argumentation schemes, keeping the essence of the argumentation schemes as devised by Walton [89] and reflecting natural (human-like) argumentation, in such practical frameworks and in particular in agent-oriented programming languages, which will allow us to implement multi-agent systems empowered by argumentation techniques in a more principled way. This is so because, firstly, Walton's approach provides an elegant model for knowledge engineering in multi-agent systems based on argumentation schemes, which seems an essential step towards developing multi-agent system applications that take advantage of argumentation schemes. Secondly, because Walton's approach provides a refined and modular mechanism for agents to reason and communicate using argumentation schemes, reflecting the essence of how humans argue. Also, our approach allows implicit information in argumentation schemes to be kept implicit at the implementation level, whereas other practical approaches explicitly represent them as additional premises or need to break an argumentation scheme into separate arguments.

In this work, we propose a novel argumentation framework, formally defined and implemented, that takes into consideration the general structure of argumentation schemes and agent-oriented programming languages as its core. Thus, after modelling (or importing) argumentation schemes into our framework, agents are directly able to construct, communicate, and reason with arguments instantiated from those argumentation schemes, also addressing important properties when using argumentation schemes, e.g., scheme awareness [91]. This work extends our previous paper [50], in which we took the initial steps towards the development of this framework. In this paper, we not only present the framework, but we also describe in detail how it can be used to implement argumentation-based reasoning and argumentation-based dialogues using argumentation schemes, presenting various examples of their use.

The main contributions of this work are: (i) we propose an argumentation framework that has the general structure of argumentation schemes as its core. Using that general structure, we can represent argumentation schemes of different levels of specificity, including those in which there is implicit information pointed out by critical questions; our framework maintains the essence of argumentation schemes as defined by Walton [89], in which it is the matching between the argument and its reasoning pattern (argumentation scheme) that brings to light the relevant implicit information necessary to judge the validity of an argument; (ii) we show how our framework can be used to implement argumentation-based reasoning mechanisms based on argumentation schemes, in which agents first instantiate and evaluate each argument individually using its respective argumentation scheme and then, considering the *acceptable instances* of arguments, agents define the set of *acceptable arguments* given a particular argumentation semantics; (iii) we experimentally evaluate the performance of our implementation for argumentation-based reasoning considering different levels of specificity for argumentation schemes, i.e., using single or chained argumentation schemes, varying the number of premises and critical questions. Our results confirm our hypothesis that the more complex the reasoning needed (using chained argumentation schemes), the longer it takes for agents to reach a conclusion. Also, our results provide guidance to the process of knowledge engineering for argumentation schemes in multi-agent systems using our approach, depending on the application's needs, i.e., detailed reasoning versus faster reasoning; (iv) we show how our framework can be used to implement argumentation-based dialogues based on argumentation schemes. We propose a dialogue protocol in which agents can question the argumentation scheme used to instantiate arguments when they are not aware of that argumentation scheme, considering the problem of *scheme awareness* in argumentation-based dialogues [91]. We implemented the proposed protocol in an agent-oriented programming language and we show the outputs generated from different argumentation-based dialogues, considering different lines of argumentation; and (v) we provide an open-source implementation of our framework in the Jason multi-agent programming platform [12].

This paper is organised as follows. In Section 2, we present the background of this work, giving an overview of how argumentation techniques have been used in artificial intelligence and multi-agent systems, also introducing the notion of argumentation schemes. In Section 3, we introduce our argumentation-scheme-centred framework and formally define it. In Section 4, we describe our approach to argumentation-based reasoning using argumentation schemes, presenting examples and an empirical evaluation of our implementation. In Section 5, we describe our approach to argumentation-based dialogues using argumentation schemes, introducing a dialogue protocol based on argumentation schemes, and showing various examples of dialogues generated according to the implementation of that protocols. In Section 6, we discuss the main properties of our framework. In Section 7, we describe some work related to our approach. In Section 8, we conclude the paper also pointing out future directions for our research.

2. Background

2.1. *Argumentation in artificial intelligence and multi-agent systems*

Computational argumentation has been largely studied as an abstract formalism [19], which provides important insights into the nature of argumentation. In abstract argumentation, arguments are represented as atomic entities and they have no internal structure. Thus, arguments and attacks between arguments are left unspecified. This abstract perspective provides some advantages in studying argumentation [10], in which it is possible to consider only the semantic level, i.e., given a set of arguments and an unspecified attack relation it is possible to analyse which arguments are acceptable in that set/context/perspective.

When we need a more detailed formalisation than abstract argumentation, regarding the structure of arguments and the nature of attack relations, we need to adopt a formalism based on *structured argumentation*, which specifies a computational model of argument [10]. In structured argumentation, a formal language is used to represent knowledge, and arguments and counterarguments can be constructed from that knowledge. Arguments are structured in the sense that premises and conclusion of arguments are made explicit, and the relationship between premises and conclusion is formalised, normally using logical entailment. Attacks between arguments than can be formalised identifying conflicting information in their structure. The best known structured argumentation frameworks are ASPIC+ [42], Assumption-Based Argumentation (ABA) [80], Defeasible Logic Programming (DeLP) [26], and Deductive Argumentation [11].

Studies in computational models of argument can be classified into two classes: *monological argumentation* and *dialogical argumentation* [10]. Monological argumentation has an emphasis on analysing a set of arguments and counterarguments (generated from some knowledge or a neutral third party), evaluating them in order to define which ones are acceptable and, consequently, making decisions based on the conclusions of those acceptable arguments. Dialogical argumentation has an emphasis on the exchange of arguments and counterarguments between agents, including not only how arguments are generated and evaluated (i.e., monological argumentation) but also how agents interact, how agents change their beliefs, protocols, and strategies that agents can adopt in argumentation-based dialogues.

In artificial intelligence, particularly in multi-agent systems, *monological argumentation* is used to implement reasoning mechanism for agents, in which agents are able to execute reasoning over uncertain and conflicting information, reaching well supported conclusions, and consequently, well supported decisions [17,47,65]. On the other hand, *dialogical argumentation* is used to design and implement argumentation-based communication between agents. Using a variety of types of dialogues, for example those described in [88], agents are able to explain their positions in a deliberation dialogue, to persuade their opponents in a negotiation, and so forth [4,33,38,58,59]. Normally, approaches to dialogical argumentation have a direct dependence on some monological argumentation framework, given that agents first need to construct and evaluate arguments from the knowledge available to them (their knowledge bases and arguments from others agents), then they can communicate arguments in argumentation-based dialogues. Also, some approaches use a third party in argumentation-based dialogues in order to describe the current "winning" arguments in that dialogue; this agent executes monological argumentation over the other agents' arguments, for example, as in [62].

The history of development, from abstract argumentation to structured argumentation, have made the field of computational argumentation mature in theory, and a few approaches to applying argumentation-based techniques in multi-agent systems started to appear in practice. Most of this practical work focuses on using an argumentation scheme (reasoning pattern) as the core component of the argumentation mechanism. Normally, the argumentation scheme used is specified for particular applications, modelling the needs of that application domain. For example, in [81] argumentation schemes have been specified for analysing the provenance of information, in [57] argumentation schemes have been specified for reasoning about trust, in [77] argumentation schemes have been specified for arguing about transplantation of human organs, in [46] argumentation schemes have been specified for implementing data access control between smart applications, and so forth. Thus, we believe that argumentation schemes can play an important role in the development of applications that take advantage of argumentation-based techniques, as the literature also suggests.

However, even though some authors (e.g., [48,63,70]) have suggested that argumentation schemes could be represented into such structured argumentation frameworks, for example using defeasible inferences rules, there is no formal and general framework that takes into account the particular structure of argumentation schemes in multi-agent systems, e.g., capturing the role of the implicit critical questions associated with each argumentation scheme. Similar claims that reinforce the missing computational representation for argumentation schemes can be found in papers by others, for example in [7]: "Douglas Walton's influential argumentation schemes remain an important insight, one which needs to be embraced by computational modelling of argument. His work comes from informal logic and was designed for manual analysis of argumentation. For computational purposes much detailed work remains to be done before the use of argumentation schemes can be as standard and well understood as logical deduction" [7].

In this work, we propose a formal argumentation framework in multi-agent systems in which argumentation schemes are the core of the framework. Our approach considers a general structure for representing argumentation schemes (with or without implicit critical questions).

2.2. Argumentation schemes

Argumentation schemes are patterns for arguments (or inferences) representing the structure of common types of arguments used both in everyday discourse as well as in special contexts such as legal and scientific argumentation [87,89]. Besides the familiar deductive and inductive forms of arguments, argumentation schemes represent forms of arguments that are *defeasible*.[1] This means that an argument may not be strong by itself (i.e., it is based on disputable inferences), but it may be strong enough to provide evidence that warrants rational acceptance of its conclusion [83]. Conclusions from argumentation schemes can be inferred in conditions of uncertainty, lack of knowledge, lack of resources, or insufficient time. This means that the reasoner must remain open-minded to new pieces of evidence that can invalidate previous conclusions [89]. These circumstances of uncertainty and lack of knowledge, resources, or time are, inevitably, characteristics of multi-agent systems, which deal with dynamic environments and organisations [92]. Thus, argumentation schemes have a natural application in multi-agent systems.

Many arguments include non-explicit (implicit) conditional premises or warrants, linking the explicit premises to the conclusion. Critical questions pointed by the argumentation scheme used to instantiate an argument are used to reveal such implicit information, playing an important role shifting the dialogue (or reasoning) to the revealed information [87]. Thus, the acceptance of a conclusion from an instantiation of an argumentation scheme is directly associated with so-called *critical questions*, which may be asked before the default conclusion from an argument (labelled by an argumentation scheme) is accepted. Together, the argumentation scheme and the matching set of critical questions are used to evaluate a given argument in a particular case, considering the context in which the argument occurred [89].

Arguments instantiated from argumentation schemes and properly evaluated utilising their critical questions can be used by agents in their reasoning and communication processes. In both situations, other arguments, probably instantiated from other argumentation schemes, are compared in order to arrive at an acceptable conclusion. After an argument is instantiated from an argumentation scheme and evaluated by its set of critical questions, the process follows the same principle of any argumentation-based approach, where arguments for and against a point of view (or just for an agent's internal decision making) are compared until eventually arriving at a set of acceptable arguments.

To exemplify our approach, we adapted the argumentation schemes *Argument from Position to Know* from [87] to the context of a multi-agent (organisational) platform, so that we are able to refer to concepts present in multi-agent systems, for example roles that agents play in the system can be referred to within the scheme (it enables a clear reference to concepts of multi-agent systems). Consider the *Argument from role to know in multi-agent systems* [48] (*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?

[1]Sometimes called *presumptive*, or *abductive* as well.

The argumentation scheme from role to know (and the argumentation scheme from position to know as well) exemplify some aspects of argumentation schemes pointed out by Walton in [87]:

"*argumentation schemes have more of a rhetoric or persuasive function than a logical function*"

in which the position/role and trustworthiness of the individual from where the information comes has an important weight in evaluating an argument instantiated from those schemes. Instantiating this argumentation scheme means replacing the variables Ag, R, A and S with some atomic formulas from the application domain, as we will see in the next section.

3. An argumentation-scheme-centred argumentation framework

Some approaches in the argumentation literature have suggested that argumentation schemes [87,89] could be translated into defeasible inferences [48,49,63], and the acceptability of arguments, instantiated using these rules, could be used to extend frameworks such as ASPIC+ [43], DeLP [27], and others [47, 53]. However, those approaches do not take into consideration that most of the critical questions related to the argumentation schemes are not explicitly represented in the argument, as clearly argued by Walton [87]. Thus, those approaches would require to make the critical questions related to the scheme used to instantiate that argument explicit in the representation of that argument; that is, it would be necessary to include an explicit representation of all critical questions (usually representing presumptions and/or exceptions for the application of that scheme [66]), as premises of the argument, or even modelling those critical questions using other arguments undercutting the main argument (i.e., an argument saying that defeasible inference rule does not apply in that particular case) [64]. Otherwise, when such critical questions are not represented as part of the argument, and an agent is not able to infer from which argumentation scheme that argument comes from, all that information is lost.

Although approaches that explicitly represent critical questions on arguments do not face the problem of losing information, we believe they do not embrace the essence of argumentation schemes[2] discussed by Walton [89], in which arguments are instances of reasoning patterns (argumentation schemes) and it is this link (identifying the argumentation scheme used to instantiate an argument) that bring to light the critical questions applying to an argument. We also believe it is essential to keep this essence when applying argumentation in dialogues between agents and between agents and humans.

In this work, we present an argumentation framework in which argumentation schemes are the core of our framework. We propose a computational representation for argumentation schemes, and of arguments instantiated from those reasoning patterns, that overcomes the problem of making explicit information that should be implicit in an argument. Moreover, we describe how agents may reason and communicate using our proposed structure of arguments.

In order to make explicit the representation of arguments, we introduce the language L used in this work. We use a first-order language as the basis for our representation, given that most agent-oriented programming languages are based on logic programming, and we are interested not only in the formal specification of our framework but also in its implementation. In particular, we have a set of atomic formulæ $\{p_0, \ldots, p_n\} \in L$, and a set of defeasible inference rules $\{(p_i, \ldots, p_j \Rightarrow p_k), \ldots, (p_l, \ldots, p_m \Rightarrow p_o)\} \in L$. 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 predicate symbols – e.g.,

[2]Extensively used in other areas, for example teaching critical thinking skills.

`john`, `doctor` and `role(john,doctor)`. We use “$\neg$” to represent strong negation in L, e.g., $\neg$`reliable(pietro)` means that `pietro` is *not* reliable. We also use *negation as failure* “`not`”, e.g., `not(reliable(pietro))` means that an agent does not know if `pietro` is reliable. Atomic formulæ, sets of atomic formulæ, and defeasible inference rules can be annotated with relevant information used in the inference mechanism. In this paper, annotations are ground literals representing the names of argumentation schemes, e.g., $p_{l_{[sn]}}$ and $\{p_i \wedge \cdots \wedge p_j\}_{[sn]}$ 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 to the inference modelled by the argumentation scheme *sn*.

Further, all knowledge available to an agent `ag` is represented as $\Delta_{\mathtt{ag}}$. When an agent `ag` is aware of some information ψ (i.e., ψ is part of or can be inferred from its belief base), we write $\Delta_{\mathtt{ag}} \models \psi$, with $\psi \in L$. For example, when an agent `ag` is aware of a defeasible inference rule $(p_l, \ldots, p_m \Rightarrow p_o)$ we write $\Delta_{\mathtt{ag}} \models (p_l, \ldots, p_m \Rightarrow p_o)$. For example, $\Delta_{\mathtt{ag}} \models$ (`penguin(X)` $\Rightarrow \neg$`flies(X)`) means that agent `ag` is aware of the defeasible inference stating that *presumably, all penguins do not fly*.

We introduce the formal definition for argumentation schemes used in this paper as follow:

Definition 1 (Argumentation scheme)**.** Formally, an argumentation scheme is a tuple $\langle sn, C, P, CQ\rangle$ with *sn* the argumentation scheme identifier (i.e., its name), C the conclusion of the argumentation scheme, P the premises, and CQ the associated critical questions.

Considering the logic language L, an argumentation scheme $\langle sn, C, P, CQ\rangle$ is represented using a (not ground) defeasible inference rule of the type $(p_i, \ldots, p_j \Rightarrow p_k)_{[sn]}$, in which $\{p_i, \ldots, p_j\} = P$ and $p_k = C$. Note that the inference rule is annotated[3] with the name of the argumentation schemes *sn*, which is used to refer to the critical questions associated with that schemes. The critical questions related to the scheme are (not grounded) formulæ of the type $\{p_{0_{[sn]}}, \ldots, p_{n_{[sn]}}\} \in L$, with $\{p_{0_{[sn]}}, \ldots, p_{n_{[sn]}}\} = CQ$.

For example, the argumentation scheme *role to know* can be represented using the following defeasible inference rule:[4]

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

with the argumentation scheme name $sn =$ `role_to_know`, the conclusion $C =$ `Conclusion`, and premises $P = \{$`role(Agent,Role)`, `role_to_know(Role,Domain)`, `asserts(Agent, Conclusion)`, `about(Conclusion,Domain)`$\}$.

The associated critical questions CQ are:

- `role_to_know(Role,Conclusion)`$_{\mathtt{[as(role\_to\_know)]}}$.
- `reliable(Agent)`$_{\mathtt{[as(role\_to\_know)]}}$.
- `asserts(Agent,Conclusion)`$_{\mathtt{[as(role\_to\_know)]}}$
- `role(Agent,Role)`$_{\mathtt{[as(role\_to\_know)]}}$.

Definition 2 (Argument)**.** An argument is a tuple $\langle S, c\rangle^{\theta}_{\mathtt{sn}}$, where $\langle \mathtt{sn}, P, C, CQ\rangle$ is the argumentation scheme used to instantiate that argument, θ is a most-general unifier for the premises in P and the agent's current beliefs, 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 P – i.e., for all

[3]We took inspiration from Labelled Deductive Systems (LBS) [24,25] to model annotated formulæ.

[4]Recall that an argumentation scheme is a non grounded reasoning pattern.

$p \in P$, $p\theta \in S$ – and the inference rule corresponding to the scheme ($P \Rightarrow C$); the conclusion c is the instantiation $C\theta$ such that $S \models c$ (c can be inferred from S).

For example, considering the argumentation scheme *role to know*, imagine that an agent $\mathtt{ag}$ knows that $\mathtt{john}$ (another agent in the system) is playing the role of $\mathtt{doctor}$ – $\mathtt{role(john, doctor)}$ – within the organisation of the multi-agent system. Further, $\mathtt{ag}$ knows that doctors know about cancer – $\mathtt{knows(doctor, cancer)}$. Therefore, if $\mathtt{john}$ asserts that "*smoking causes cancer*" – $\mathtt{asserts(john, causes(smoking, cancer))}$, and given that causes of cancer are a subject matter related to cancer – $\mathtt{about(causes(smoking, cancer), cancer)}$, $\mathtt{ag}$ is able to instantiate the argumentation scheme *role to know*, which allows $\mathtt{ag}$ to conclude that smoking causes cancer – $\mathtt{causes(smoking, cancer)}$.

Note that an agent can instantiate different arguments, say $\langle S_1, c_1\rangle^{\theta_2}_{\mathtt{sn}}$ and $\langle S_2, c_2\rangle^{\theta_1}_{\mathtt{sn}}$, from the same argumentation scheme $\mathtt{sn}$, when $\theta_1 \neq \theta_2$. Also, an agent is able to use the same argumentation scheme to instantiate conflicting arguments. Following our example, imagine that another agent also (presumably) playing the role of doctor, called $\mathtt{pietro}$, asserts that "*smoking does not cause cancer*" – $\mathtt{asserts(pietro, \neg causes(smoking, cancer))}$. Any agent aware of both assertion, John's and Pietro's assertions, is able to construct conflicting arguments for $\neg\mathtt{causes(smoking, cancer)}$ and $\mathtt{causes(smoking, cancer)}$ (as we will see later, both attacking each other). However, the agents are able to question whether $\mathtt{john}$ and $\mathtt{pietro}$ are reliable (trustworthy) sources, if they really play the role of $\mathtt{doctor}$, and the other questionable points indicated by the critical questions in the argumentation scheme used in order to check the validity of that particular conclusion. Here, an important issue on the representation of arguments appears, as not all critical questions (for example the reliability of an agent) refer to information explicitly represented in the structure of arguments instantiated from argumentation schemes. As described in [84], the critical questions of an argumentation scheme play an important role in: (i) criticising the scheme's premises; (ii) pointing out exceptional situations in which the scheme should not be used; (iii) representing conditions for a scheme's use (for example, a particular context). Thus, it is essential to consider the critical questions when evaluating an individual instance of argument from an argumentation scheme, and when the critical questions are not positively answered, that argument instance might not be acceptable.[5] For example, imagine that an agent has the information that "*Pietro is not a reliable source*" – $\neg\mathtt{reliable(pietro)}$. In that case, that agent is not able to answer positively the critical question $\mathtt{reliable(pietro)}$, thus it is rational to think that instance of the argumentation scheme (i.e., that argument) might not be acceptable for that agent; the argument concluding $\neg\mathtt{causes(smoking, cancer)}$ might not be an acceptable instance of the argumentation scheme *role to know* for such a rational agent.

Definition 3 (Acceptable instance of an argument from an argumentation scheme)**.** An argument $\langle S, c\rangle^{\theta}_{\mathtt{sn}}$, instantiated from an argumentation scheme $\langle \mathtt{sn}, P, C, CQ\rangle$, is an *acceptable instance of that argumentation scheme* to an agent *ag* (with $\Delta_{\mathtt{ag}}$ its knowledge base) iff: (i) for all premises $p \in S$, $\Delta_{ag} \models p\theta$ either because p is asserted in its knowledge base, or because p can be inferred from Δ_{ag} (possibly using other acceptable instances of argumentation schemes); and (ii) all critical questions related to the argumentation scheme $\langle \mathtt{sn}, C, P, CQ\rangle$ are positively answered by *ag*, that is, $\forall Cq_i \in CQ$, $\Delta_{ag} \models Cq_i\theta$.

[5]Different agent profiles can be explored, for example, those presented in [58,60]; they will be considered in our future work when applying our framework in argumentation-based dialogue. When using our framework for argumentation-based reasoning, it makes sense an agent to evaluate such arguments being able to answer all critical questions related to them.

Note that instantiating an argumentation scheme requires an agent checking the acceptability of the premises used in that argument, this automatically refers to the agent checking attacks against the premises of that argument instance.[6] Also, positively answering the critical questions related to the argumentation schemes used to instantiate that argument requires instantiating the critical questions. Considering the language introduced above, we are able to model critical questions that cover all roles mentioned by [84], for example, for some scenarios it would be interesting to model critical questions pointing out to the absence of a particular piece of information – $\texttt{not}(\neg\texttt{reliable}(\texttt{pietro}))$ (the agent does not have the information that Pietro is not reliable). When the application requires more complex argumentation (inferences) regarding critical questions, predicates representing critical questions may be the conclusion of other arguments, which will be instantiated from argumentation schemes with their own critical questions, as we show later in this paper.

After instantiating arguments from the argumentation schemes available to it, an agent needs to check if those arguments result in conflicts. Considering a dialectical point of view, two types of attack (conflict) between arguments can be considered, according to [89]: (i) a strong kind of conflict, where one party has a thesis to be proved, and the other party 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 party doubts that thesis, but has no opposing thesis of their 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. When considering the computational model of arguments, this difference between conflicts is inherent from the structure of arguments and can be found in the work of others (e.g., [63]). Both types of conflicts are also considered in monological argumentation frameworks. On the one hand, the stronger kind of conflict refers to 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, that is, an argument does not imply that the conclusion of another argument is false, but it implies that some information used as support of that conclusion (e.g., some piece of information in the support of another argument) is not true.

Definition 4 (Conflicting information)**.** Two pieces of information φ and ψ are said to be in conflict with each other when:

- ψ and φ are the negation of each other, e.g., $\psi = \texttt{causes}(\texttt{smoking}, \texttt{cancer})$ and $\varphi = \neg\texttt{causes}(\texttt{smoking}, \texttt{cancer})$.
- ψ and φ are semantically declared as conflicting information, i.e., $\psi = \texttt{reliable}(\texttt{john})$ and $\varphi = \texttt{unreliable}(\texttt{john})$, with some declaration of complement, conflict, or contrary information, e.g., $\texttt{complement}(\texttt{reliable}(\texttt{X}), \texttt{unreliable}(\texttt{X}))$.

For both cases we adopt a general operator for conflict $\overline{\psi}$, in which $\varphi \equiv \overline{\psi}$ means that ψ and φ are conflicting information. It follows that $\neg\phi \equiv \overline{\phi}$ and $\phi \equiv \overline{\neg\phi}$.

Attacks between arguments are identified by looking at their structures and finding conflicting information, as formalised below.

Definition 5 (Attacks between arguments)**.** An argument $\langle S_1, c_1\rangle^{\theta_1}_{\texttt{sn}_\texttt{i}}$ attacks another argument $\langle S_2, c_2\rangle^{\theta_2}_{\texttt{sn}_\texttt{j}}$, when:

[6]In ASPIC+ for example, attacks against the premises are described as *undermine attacks* [42].

- c_1 is in conflict with c_2, i.e., $c_1 \equiv \overline{c_2}$.
- c_1 is in conflict with some part of S_2, i.e., $c_1 \equiv \overline{\psi}$, with $\psi \in S_2$.

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

For example, considering an agent ag and its knowledge base Δ_{ag}, ag is able to construct two different arguments using the argumentation scheme *role to know*; both arguments are in conflict with each other at their conclusions:

```
⟨{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)},
 causes(smoking,cancer)⟩[as(role_to_know)] .
```

```
⟨{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)},
 ¬causes(smoking,cancer)⟩[as(role_to_know)] .
```

Note that both arguments above are acceptable instances of the argumentation scheme *role to know*, according to Definition 3. However, when defining the acceptability of an argument (semantically), it is not enough to check if it is an acceptable instance of an argumentation scheme; it is necessary to check if that argument survives attacks from other acceptable instances of argumentation schemes. At this point, it would be reasonable to conclude that both arguments are not (semantically) acceptable to ag, and it would not be able to conclude either causes(smoking, cancer) or ¬causes(smoking, cancer). However, imagine that pietro is not a doctor, i.e., role(pietro, doctor) is not true (it was a misperception of ag), and ag becomes aware of that information through the hospital director named mathew.

$$\Delta_{ag} = \left\{ \begin{array}{l} \texttt{reliable(mathew)} \\ \texttt{asserts(mathew, role(pietro, cleaning\_staff))} \\ \texttt{role(mathew, hospital\_director)} \\ \texttt{role\_to\_know(hospital\_director, employees)} \\ \texttt{about(role(pietro, cleaning\_staff), employees).} \end{array} \right\}$$

Given the knowledge that ag has about the hospital, partially shown above, in Δ_{ag}, it is able to construct an argument using the argumentation scheme *role to know* concluding that pietro is a cleaning staff and not a doctor:

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

Thus, the argument concluding ¬causes(smoking, cancer) is not an acceptable instance of the argumentation scheme *role to know* anymore, considering that role(pietro, cleaning_staff) is conflicting information with role(pietro, doctor) (semantically declared at ag knowledge base). At this point, it would be reasonable for ag to conclude causes(smoking, cancer).

Definition 6 (Acceptable arguments)**.** An argument $\langle S_1, c_1\rangle^{\theta}_{\mathtt{sn_i}}$ is acceptable to an agent if it is an acceptable instance of the argumentation scheme $\mathtt{sn_i}$, according to Definition 3, and either it is not attacked by any other acceptable argument instance or, when attacked, the attacking argument is itself attacked by an *acceptable* argument. That is, considering the set of arguments an agent is able to construct from its knowledge base, an acceptable argument is either not attacked by any other argument, or when it is attacked by another argument, that other argument is attacked by some other acceptable argument.

Considering the three arguments above, the argument concluding that causes(smoking, cancer) is attacked by the argument concluding ¬causes(smoking, cancer), which is attacked by the argument concluding role(pietro, cleaning_staff). Considering that the argument concluding role(pietro, cleaning_staff) is not attacked by any other argument, role(pietro, cleaning_staff) and causes(smoking, cancer) are the two acceptable conclusions (and their respective arguments) in our example. We write $\Delta_{\mathtt{ag}} \models \langle S_1, c_1\rangle^{\theta}_{\mathtt{sn_i}}$ to represent when $\langle S_1, c_1\rangle^{\theta}_{\mathtt{sn_i}}$ is an acceptable argument for ag. Also, we write $\Delta_{\mathtt{ag}} \models c_1$ to represent that c_1 is an acceptable conclusion for ag.

4. Argumentation-based reasoning using argumentation schemes

One characteristic we desire in our framework is *generality*. The generality of our framework to represent argumentation schemes has been evaluated in Panisson's PhD thesis [44], in which 11 argumentation schemes (with and without implicit critical questions) from different areas in the literature have been specified using our framework, evaluated, and implemented in an agent-oriented programming language. Here, starting from Section 4.1, we focus on demonstrating examples of how our framework for argumentation schemes can be used to implement argumentation-based reasoning in multi-agent systems. Also, in Section 4.2, we evaluate our implementation in order to demonstrate how fast agents are able to reach a particular conclusion supported by arguments instantiated from argumentation schemes using our implementation, investigating how much the number of premises and critical questions influence the time necessary for agents to reach such a conclusion.

4.1. Representing and reasoning with argumentation schemes

It is important to mention that in our approach we are able to specify argumentation schemes for agents with different levels of bounded rationality, depending on the application needs. This is an important characteristic of our approach, given that different applications might require different levels of specificity for argumentation schemes [84]. The most common approach when applying argumentation schemes in multi-agent systems is to consider only one argumentation scheme without chained arguments. Examples of this kind of argumentation schemes are found in [28,34,57,79,81]. In all those cases, during reasoning, agents check if they are able to construct acceptable instances of arguments from the argumentation schemes, according to Definition 3, used in a particular application domain. When the application domain does not require chained arguments, then premises and critical questions related to an argument are directly asserted (or omitted from) agents' belief bases. Thus, after constructing acceptable instances of arguments, an agent will check for conflicts between the conclusions of arguments,[7] determining the acceptable ones, according to Definition 6. When the application domain requires chained arguments (e.g., as in [46]), agents check if they are able to infer the premises and critical questions of a particular argument, according to Definition 3, then constructing acceptable instances of arguments from the argumentation schemes available to them. Thus, agents check the acceptability of those instances of arguments, checking both kinds of attacks, given that premises can be inferred from arguments and those arguments can be attacked by other arguments.

Consider the *argumentation scheme from role to know*, extensively used as an example in this paper; its representation was introduced in Section 2.2 followed by a few examples demonstrating how agents are able to reason using arguments instantiated from it. In the first example given, *ag* uses the argumentation scheme role to know to instantiate arguments for and against the conclusion that "*smoking causes cancer*". In the second example, the agent *ag* uses the same argumentation scheme to construct an argument supporting that Pietro is a cleaner, which then is used to attack the argument concluding that "*smoking does not cause cancer*" based on Pietro's assertions, because now *ag* does not believe Pietro is in a role to know about the subject of cancer. Here, we extend those examples to demonstrate when agents are also able to construct arguments supporting/attacking critical questions of other arguments using this single argumentation scheme.

Imagine a scenario in which an agent $\mathtt{ag}$ needs to make a decision about $\mathtt{quitting\ smoking}$ or not, and it will make that decision based on information about whether smoking causes cancer or not. That is, the agent will quit smoking if it concludes that smoking causes cancer. In our scenario, *ag* will reason about that conclusion based on Pietro's and John's assertions (assuming that both are doctors).

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

However, initially $\mathtt{ag}$ does not know if John and Pietro are reliable, $\mathtt{reliable(john)}$ and $\mathtt{reliable(pietro)} \notin \Delta_{ag}$. Thus, $\mathtt{ag}$ asks other agents (we will call those agents reliability advisers)

[7]We are considering that argumentation schemes are consistently modelled, and they do not allow circular arguments [90].

whether Pietro and John are reliable or not, executing a broadcast message and receiving the following assertions from Ana: $\mathtt{asserts(ana, reliable(john))}$, $\mathtt{asserts(ana, \neg reliable(pietro))}$, Jana: $\mathtt{asserts(jana, reliable(pietro))}$, $\mathtt{asserts(jana, \neg reliable(john))}$, and Carlo: $\mathtt{asserts(carlo, \neg reliable(pietro))}$, $\mathtt{asserts(carlo, reliable(john))}$.

$$\Delta_{ag} = \left\{ \begin{array}{lr} \mathtt{asserts(ana, \neg reliable(pietro))} & \mathtt{asserts(ana, reliable(john))} \\ \mathtt{asserts(jana, reliable(pietro))} & \mathtt{asserts(jana, \neg reliable(john))} \\ \mathtt{asserts(carlo, \neg reliable(pietro))} & \mathtt{asserts(carlo, reliable(john))} \\ \mathtt{role(ana, reliability\_adviser)} & \mathtt{reliable(ana)} \\ \mathtt{role(jana, reliability\_adviser)} & \mathtt{\neg reliable(jana)} \\ \mathtt{role(carlo, reliability\_adviser)} & \mathtt{reliable(carlo)} \\ \mathtt{role\_to\_know(reliability\_adviser, reliability)} & \\ \mathtt{about(reliable(\_), reliability)} & \\ \mathtt{about(\neg reliable(\_), reliability)} & \end{array} \right\}$$

Considering the assertions made by Ana, Jana, and Carlo (the reliability advisers), and also considering that $\mathtt{ag}$ believes Ana and Carlo are reliable advisers and Jana is not a reliable adviser, as shown in the knowledge base above, $\mathtt{ag}$ concludes that John is reliable, $\mathtt{reliable(john)}$, and that Pietro is not reliable, $\mathtt{\neg reliable(pietro)}$. Both conclusions are supported by acceptable arguments, according to Definition 6, which are first built as acceptable instances of the argumentation scheme *role to know*, according to Definition 3. While Ana's and Carlo's assertions are used to instantiate arguments that are acceptable instances of the argumentation scheme role to know, and they are used to support information used to help answering critical questions by $\mathtt{ag}$, Jana's assertions are ignored because Jana is not reliable and then $\mathtt{ag}$ is not able to build acceptable instances of arguments from the argumentation scheme *role to know* using her assertions, according to Definition 3, i.e., $\mathtt{ag}$ is not able to answer the critical question related to the reliability of Jana.

Using that information, $\mathtt{ag}$ concludes that *smoking causes cancer*, given it has an acceptable argument supporting $\mathtt{causes(smoking, cancer)}$ based on John's assertion, also considering that John is a reliable source of information, which is supported by arguments built using Carlo's and Ana's assertions, who also are reliable. On the other hand, the information that *smoking does not cause cancer*, $\mathtt{\neg causes(smoking, cancer)}$, is not supported by an acceptable argument, given that $\mathtt{ag}$ is not able to construct an acceptable instance of argument using the argumentation scheme *role to know* based on Pietro's assertion, according to Definition 3, because it concludes that Pietro is not a reliable source of information, which is supported by the arguments built using Ana's and Carlo's assertions.

4.2. Experimental results

We have implemented our approach[8] extending the argumentation-based framework reported in [47, 53], which is implemented in the Jason [12] multi-agent platform. We have evaluated our implementation to check how much time an agent needs to construct an acceptable instance of an argument from an argumentation scheme, depending on the number of premises and critical questions present in such scheme. In our experiments, we have considered both approaches for modelling argumentation schemes, i.e., with and without chained arguments.

First, we evaluate our implementation when modelling argumentation schemes without chained arguments, in which premises and critical questions are asserted into agents' belief base. Figure 1 shows our

[8]The implementation is available in https://github.com/AlisonPanisson/ASinMAS.

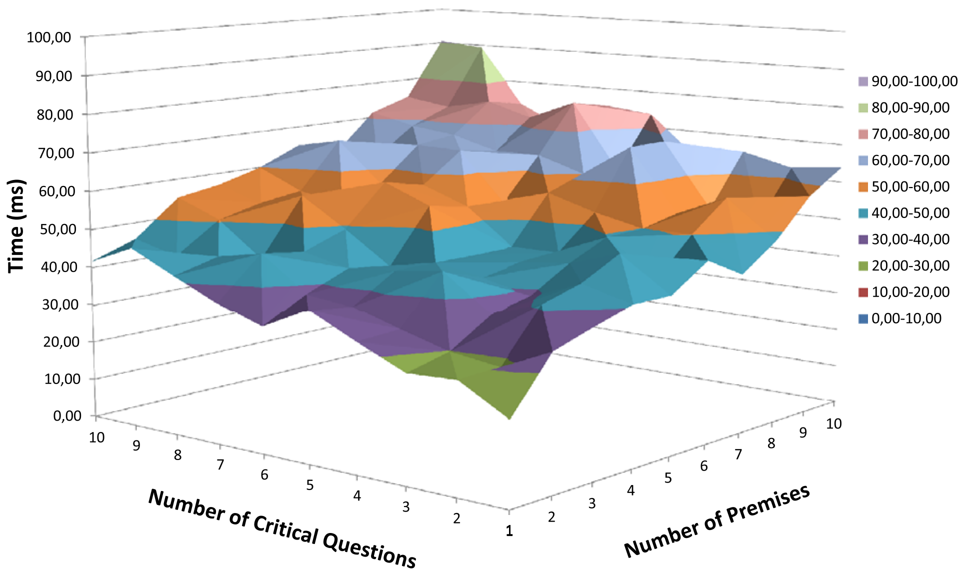


Fig. 1. Time (ms) for reaching a conclusion considering one level of inference.

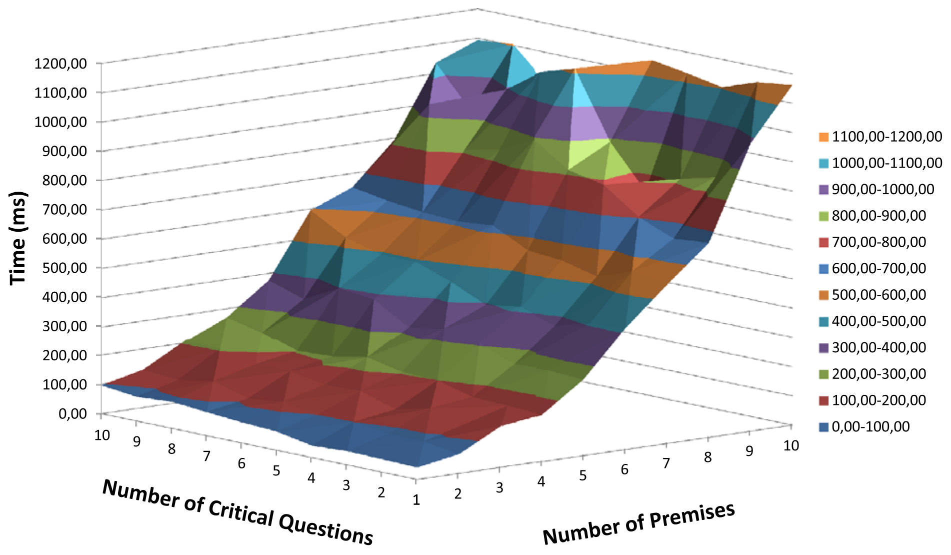


Fig. 2. Time (ms) for reaching a conclusion considering 2 levels of inference.

results, varying the number of premises and critical questions from 1 to 10. It can be noted that there is a similar influence of the number of premises and critical questions in the final time for an agent to construct an acceptable instance of an argumentation scheme.

We then evaluate our implementation considering chained arguments, in which the premises of the main argument are conclusions of other arguments constructed from different argumentation schemes. Thus, first an agent constructs acceptable instances of arguments from their respective argumentation schemes for each premise of the main argument, and using those premises it constructs the main argument as an acceptable instance of its respective argumentation scheme. Figure 2 shows our results,

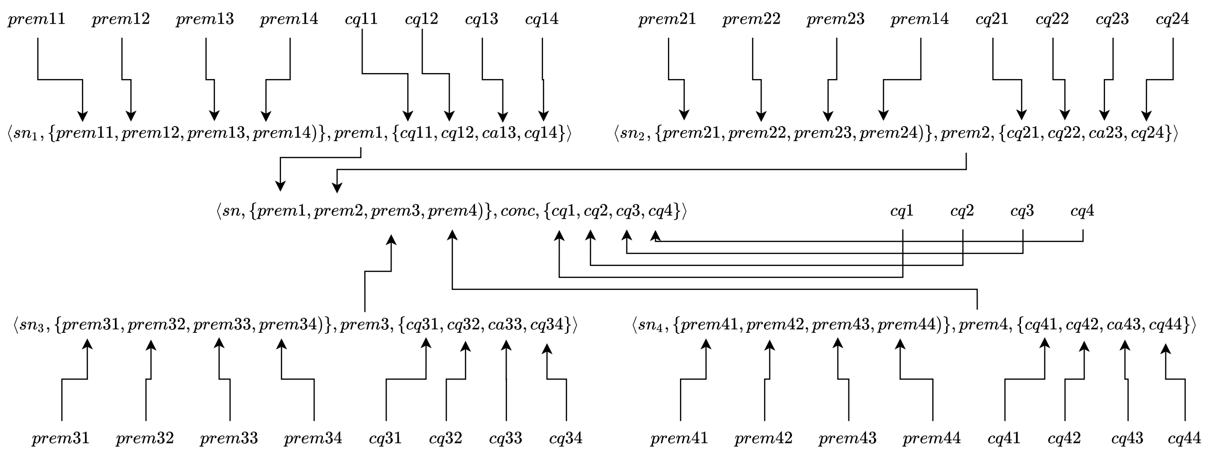

prem11
prem12
prem13
prem14
cq11
cq12
cq13
cq14
prem21
prem22
prem23
prem14
cq21
cq22
cq23
cq24
$\langle sn_1, \{prem11, prem12, prem13, prem14)\}, prem1, \{cq11, cq12, ca13, cq14\}\rangle$
$\langle sn_2, \{prem21, prem22, prem23, prem24)\}, prem2, \{cq21, cq22, ca23, cq24\}\rangle$
$\langle sn, \{prem1, prem2, prem3, prem4)\}, conc, \{cq1, cq2, cq3, cq4\}\rangle$
cq1
cq2
cq3
cq4
$\langle sn_3, \{prem31, prem32, prem33, prem34)\}, prem3, \{cq31, cq32, ca33, cq34\}\rangle$
$\langle sn_4, \{prem41, prem42, prem43, prem44)\}, prem4, \{cq41, cq42, ca43, cq44\}\rangle$
prem31
prem32
prem33
prem34
cq31
cq32
cq33
cq34
prem41
prem42
prem43
prem44
cq41
cq42
cq43
cq44

Fig. 3. Abstract inference tree generated for chained argumentation schemes with 4 premises and 4 critical questions.

varying the number of premises (and respective argumentation schemes, given each premise is the conclusion of an acceptable instance from a different argumentation scheme) and the critical questions. It can be noted that the number of premises has a greater influence on the time to construct the main argument than the number of critical questions. This results from the fact that premises of the main argument are conclusions of other arguments, which are constructed and evaluated from their respective argumentation schemes,[9] according to Definition 3, which means that they have their own critical questions to be positively answered.

Our results provide a guide for the process of knowledge engineering of argumentation schemes in multi-agent systems (specially useful when using our implementation), showing how those reasoning patterns could be modelled depending on the multi-agent application requirements. On the one hand, when defining chained argumentation schemes (generating chained arguments), which will allow agents to execute a more detailed and refined reasoning process based on argumentation, as in [46], more time is required for agents to analyse and evaluate a particular conclusion from those argumentation schemes. On the other hand, when defining a single argumentation scheme, such reasoning is simplified but faster results are reached.

Figure 3 shows an abstract example for an inference tree generated by an agent reaching a conclusion *conc* using chained argumentation schemes (i.e., all sn_i argumentation schemes are chained to the argumentation scheme *sn*) with four premises and four critical questions.

5. Argumentation-based dialogues with argumentation schemes

When participating in argumentation-based dialogues using argumentation schemes, agents must be aware of such reasoning patterns in order to have a full understanding of what is being communicated, which has been referred to as "scheme awareness" by Wells [91]. Being aware of the argumentation schemes used by others, when receiving an argument, allows agents to understand which argumentation

[9]For example, when considering chained argumentation schemes with 2 premises and 2 critical questions, besides answering the 2 critical questions of the main argumentation scheme, the premises of the main argument are the conclusion of 2 different arguments constructed from different argumentation schemes which also have 2 premises and 2 critical questions to be positively answered.

scheme has been used to instantiate that argument, identifying the critical questions related to it, being able to understand what implicit information was used by the speaker when building that argument.

Definition 7 (Scheme awareness)**.** An agent ag is aware of an argumentation scheme $\langle \mathtt{sn}, P, C, CQ\rangle$ when that argumentation scheme is in its knowledge base Δ_{ag}. We write $\langle \mathtt{sn}, P, C, CQ\rangle \in \Delta_{ag}$ to represent the fact that ag is aware of the argumentation scheme $\mathtt{sn}$.

A few approaches in the literature, for example [48,49], propose that agents should openly share such argumentation schemes in order to deal with the inherent problem of scheme awareness. An overview of those approaches to argumentation schemes in MAS is shown in Fig. 4, where all agents are able to instantiate arguments from shared argumentation schemes at runtime. Whilst those approaches solve the problem of scheme awareness as defined by Wells [91], and it is adequate to many multi-agent architectures using shared domain-specific knowledge [23,52,74], we propose a more general approach, allowing agents to have knowledge of specific argumentation schemes and to be able to communicate them when necessary.

In communication, the acceptability of arguments received from other agents is directly associated with the agents' rationality, i.e., when an agent $\mathtt{ag}$ receives an argument $\langle S, c\rangle^{\theta}_{\mathtt{sn_i}}$ from another agent, it is able to check whether or not that information is acceptable to itself, adding S to its knowledge base $\Delta_{\mathtt{ag}}$ (i.e., $\Delta_{\mathtt{ag}} = \Delta_{\mathtt{ag}} \cup S$) and checking the acceptability of c. Thus, it is essential that such an agent is aware of the argumentation scheme $\mathtt{sn_i}$ to adequately check whether that argument is an acceptable instance of that argumentation scheme according to Definition 3. Afterwards, an agent can proceed to further consider that argument looking for arguments that attack it, arguments that attack those arguments attacking the initial one, and so on, according to Definition 6.

We use $\Delta_{\mathtt{ag}} \models c$ to say that agent $\mathtt{ag}$ is able to construct an acceptable argument $\langle S, c\rangle^{\theta}_{\mathtt{sn_i}}$ according to Definition 6, with $S \subset \Delta_{\mathtt{ag}}$. Argument $\langle S, c\rangle^{\theta}_{\mathtt{sn_i}}$ refers to an acceptable instance of the argumentation scheme $\langle \mathtt{sn_i}, P, C, CQ\rangle \in \Delta_{\mathtt{ag}}$. An agent never has acceptable arguments for conflicting information, i.e., $\Delta_{\mathtt{ag}} \models c$ and $\Delta_{\mathtt{ag}} \models \overline{c}$ never both hold simultaneously. We use $\Delta_{\mathtt{ag}} \not\models c$ to mean that an agent is not

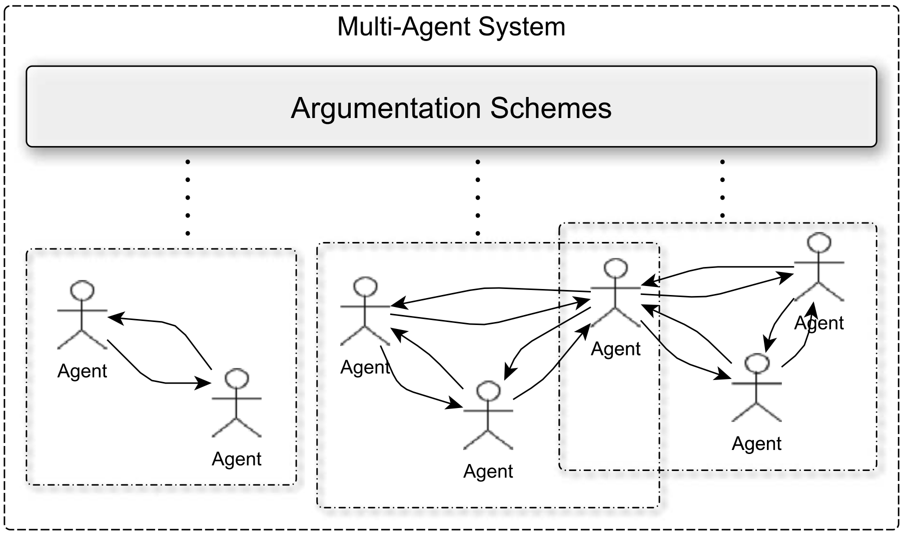


Fig. 4. Argumentation schemes shared by all agents.

able to construct an acceptable argument for c. $\Delta_{\mathtt{ag}} \not\models c$ does not imply $\Delta_{\mathtt{ag}} \models \overline{c}$; however, $\Delta_{\mathtt{ag}} \models c$ implies $\Delta_{\mathtt{ag}} \not\models \overline{c}$.

5.1. *Performatives for argumentation-based dialogues*

In the multi-agent paradigm, communication is often based on the speech-act theory [75]. In such approaches, messages have the following format $\langle \mathtt{Ags}, \mathtt{performative}, \mathtt{content} \rangle$, where `Ags` represents the agent (or set of agents) to which the message is addressed, `performative` denotes the illocutionary force of the speech act, and `content` is the (propositional) content of the message. The performatives we use in this work, and their intended meaning are informally presented below:

- `assert`: the performative `assert` is used by agents to introduce a particular claim in the dialogue. The agents receiving such a message become aware of that claim.
- `question`: the performative `question` is used by agents to question a claim, part of arguments, or a critical question related to an argument, put forward by agents in the dialogue. The `content` will be a proposition φ that either: (i) refers to a claim put forward by some agent in the dialogue; or (ii) refers to part of the support of an argument put forward by some agent in the dialogue, i.e., $\varphi \in S$ for some argument $\langle S, c \rangle^{\theta}_{\mathtt{sn}}$ that has been put forward by some agent in the dialogue; or (iii) a critical question associated to an argument that has been put forward by some agent in the dialogue, i.e., $\varphi = q\theta_{\mathtt{sn}}$, with $q \in \mathcal{CQ}$, for an argumentation scheme[10] $\langle \mathtt{sn}, \mathcal{C}, \mathcal{P}, \mathcal{CQ} \rangle$, with $\langle S, c \rangle^{\theta}_{\mathtt{sn}}$ the argument.
- `justify`: the performative `justify` is used by agents to introduce an argument in the dialogue. The content will be $\langle S, c \rangle^{\theta}_{\mathtt{sn}}$, the argument supporting c.
- `refuse`: the performative `refuse` is used by agents to refuse a previous proposition put forward by some agent in the dialogue.
- `accept`: the performative `accept` is used by agents to accept a previous proposition put forward by some agent in the dialogue.
- `question_scheme`: the performative `question_scheme` is used by agents to question which argumentation scheme has been used to instantiate an argument previously communicated during the dialogue by another agent.
- `inform_scheme`: the performative `inform_scheme` is used by agents to inform another agent about the argumentation scheme it has used to instantiate an argument previously communicated by it during the dialogue.

In the course of argumentation-based dialogues, agents make commitments based on which speech act they use. These commitments are stored in the commitment store (CS) that consists of one or more structures, accessible to all agents in a dialogue.[11] The CS is simply a subset of the knowledge base, and the union of the CSs can be viewed as the global state of the dialogue at a given time [59]. The rules that define how CSs are updated, depending on the speech act used by agents, are summarised as follows: (i) `assert`: the agent's CS is updated with the asserted content p, $CS \leftarrow CS \cup \{p\}$; (ii) `accept`: the agent's CS is updated with the accepted content p, $CS \leftarrow CS \cup \{p\}$; (iii) `question` and `refuse`: no effect over the CS; (iv) `justify`: the agent's CS is updated with the justified content contained in the set of rules and facts S (the support of the intended argument for p), $CS \leftarrow CS \cup S$;

[10]Recall that S is different from $\mathcal{P}$, considering that $\mathcal{P}$ are non-instantiated premises, and S the entire support for c.

[11]Other names are used for CS, such as *dialogue obligation store* in [40] and *dialogue store* in [73].

and (v) `question_scheme` and `inform_scheme`: no effect over the CS. Using these speech acts, various protocols can be specified.

5.2. A framework for argumentation-based dialogues

In this section, we describe a framework for the specification of dialogue games. In previous work [54], we used this framework to model argumentation-based dialogues based on a particular protocol. Here, we use the same framework to specify an argumentation-based protocol based on our approach for argumentation schemes in multi-agent systems. The framework for dialogue games is based on work by McBurney et al. [38,39], in which the elements that correspond to the dialogue game specification are:

- **Commencement Rule**: 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, an agent is allowed to use, following a particular protocol.
- **Combination Rules:** the possible combination of locutions. Normally, the choice of the next move also depends on the strategy of the agent on choosing among the allowed moves at each stage of the dialogue (e.g., corresponding the agent attitudes to assert and accept claims in the dialogue [58,60]).
- **Commitments:** 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:** when a dialogue ends.

In particular, *Dialogue rules* will govern the interactions between the agents, where each agent moves by performing one of the allowed utterances. These rules (which correspond to a dialogue game [39]) 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 [3].

Definition 8 (Dialogue game protocol [54])**.** 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 9 (Dialogue move [54])**.** 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 $\mathtt{content}$. We consider the following set of types of moves, denoted by $\mathtt{P}$ (q.v. Section 5.1): `assert`, `accept`, `refuse`, `question`, `justify`, `question_scheme` and `inform_scheme`. The content of a move (`content`) can be an argument (a set of predicates and rules), a single predicate, or and argumentation scheme.

The dialogue rules in $\mathtt{DI}$ indicate the possible moves that an agent can make following a previous move by the other agent. The formalisation we give here follows the work of Bentahar et al. [9]. To define the dialogue rules, we use a set of conditions (denoted by $\mathtt{C}$) which reflect the agents' strategies. Formally, we have:

Definition 10 (Dialogue rules [54])**.** Dialogue rules can assume one of two forms:

1) 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).

$$\bigwedge_{\substack{0<\mathtt{k}\leqslant \mathtt{n_i},\\ \mathtt{i},\mathtt{j}\in \mathtt{P}}} \left(\mathtt{M}^{\mathtt{i}}(\alpha, \beta, \mathtt{content}, \mathtt{t}) \wedge \mathtt{C}_{\mathtt{k}} \Rightarrow \mathtt{M}^{\mathtt{j}}_{\mathtt{k}}\big(\beta, \alpha, \mathtt{content}_{\mathtt{k}}, \mathtt{t}'\big)\right)$$

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

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

$$\bigwedge_{\substack{0<\mathtt{k}\leqslant\mathtt{n},\\ \mathtt{j}\in\mathtt{P}}} (\mathtt{C}_{\mathtt{k}} \Rightarrow \mathtt{M}^{\mathtt{j}}_{\mathtt{k}}(\alpha, \beta, \mathtt{content}_{\mathtt{k}}, \mathtt{t}_0)$$

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 a protocol that takes argumentation schemes under consideration. we then show how we implement that protocol (and respective dialogue rules) for Jason agents [12].

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

- We write $\Delta_{\mathtt{ag}}$ to represent agent $\mathtt{ag}$'s knowledge base. $\Delta_{\mathtt{ag}}$ includes $\mathtt{ag}$'s private knowledge and argumentation schemes.
- We write $\mathtt{CS}_{\mathtt{ag}}$ to represent agent $\mathtt{ag}$'s commitment store, which is updated according to the rules described in Section 5.1.
- We write $\Delta_{\mathtt{ag}} \models \langle S, c\rangle^{\theta}_{sn}$ to denote that an agent $\mathtt{ag}$ is able to instantiate an acceptable argument supporting c (according Definition 6) from the information available in its knowledge base $\Delta_{\mathtt{ag}}$, 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 denote that an agent ag_i is able to construct an acceptable argument supporting c from its knowledge base and ag_j's commitment store.

5.3. A protocol using argumentation schemes

Our protocol combines persuasion and information-seeking dialogues. A persuasion dialogue is used when an initiator agent believes something that it wants another agent to be convinced to believe as well [76]. In this type of dialogue, an agent starts the dialogue with an `assert` move, asserting the information (the subject of the dialogue) it wants the other agent to believe [54]. After that, 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, then closing the dialogue. However, our protocol mixes information-seeking dialogues when we allow agents to ask for the argumentation scheme used to instantiate arguments used by other agents in the dialogue.

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

In order to evaluate our framework, we have implemented the following protocol, which is an extended version of [54], making reference to components of our framework for argumentation schemes in multi-agent systems, as well as allowing agents to question the argumentation scheme used by other agents during the dialogue:

(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 make ag_j 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, $\langle S, p\rangle^{\theta}_{sn}$), checking whether it is aware of the argumentation scheme sn used to build that argument. When it is not aware of the argumentation scheme sn, it questions ag_i about the scheme executing a `question_scheme` move `question_scheme`(ag_j, ag_i, sn) (the protocol goes to (6)). Otherwise, when it is aware of the argumentation scheme used by ag_i, then it checks if the new information received from ag_i is enough to make it accept the subject of the dialogue, i.e., if it has no argument against the subject of the dialogue, considering the new 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[12] `assert`(ag_j, ag_i, $\neg p$), committing itself to provide support for $\neg p$ (the protocol goes to (2)); or (ii) it checks whether it is able to answer negatively any 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 `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 `accept`(ag_i, ag_j, p), and in the case where the accepted proposition is the subject of the dialogue, the dialogue ends with both agents believing that the subject of the dialogue holds. In the case where the accepted proposition is a critical question, the agent that accepts the critical question returns to step (4) of the protocol, accepting the subject of the dialogue, questioning another critical question, or committing itself to give support for the subject of the dialogue not being the case.
(6) an agent ag_j receives a `question_scheme` move `question_scheme`(ag_i, ag_j, sn) and answers that move by informing ag_i about the argumentation scheme sn executing a `inform_scheme` move `inform_scheme`(ag_j, ag_i, $\langle sn, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$) (the protocol goes to (7)).
(7) an agent ag_j receives an `inform_scheme` move `inform_scheme`(ag_i, ag_j, $\langle sn, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$), recalling the subject of the dialogue and then checking if it has no argument against the subject of the dialogue. 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 provide support for $\neg p$ (the protocol goes to (2)); or (ii) it checks if it is able to answer negatively any critical question related

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

to the argumentation scheme *sn* that ag_i has used to instantiate its argument. In 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 that it was able to answer negatively (the protocol goes to (2)).

5.3.1. Dialogue rules

Initial Rule: The first move (commencement rule) introduces the subject of the dialogue (where `subject(p)` means that the atomic formula `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* [18] (we use `Pr` and `Op`, respectively, to refer to them).

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

where:

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

The dialogue starts when an agent wants to argue about a given subject. The initial rule states that an agent must have an argument that supports its claim in order to start an argumentation-based dialogue (as the agent will be committed to defend 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 an argument from an argumentation scheme in $\Delta_{\mathtt{AS}}$, according to Definition 6.

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

$$\texttt{assert}(ag_i, ag_j, \texttt{p}) \wedge \mathtt{C}_{as1} \Rightarrow \texttt{accept}(ag_j, ag_i, \texttt{p})$$
$$\texttt{assert}(ag_i, ag_j, \texttt{p}) \wedge \mathtt{C}_{as2} \Rightarrow \texttt{question}(ag_j, ag_i, \texttt{p})$$

where:

$$\mathtt{C}_{as1} = \nexists \langle S, \overline{p}\rangle_{sn}^{\theta} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S, \overline{p}\rangle_{sn}^{\theta}$$
$$\mathtt{C}_{as2} = \exists \langle S, \overline{p}\rangle_{sn}^{\theta} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S, \overline{p}\rangle_{sn}^{\theta}$$

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 for its previous claim.

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

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

where:

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

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

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

$$\texttt{justify}\big(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}\big) \wedge \mathtt{C}_{js1} \Rightarrow \texttt{question\_scheme}(ag_j, ag_i, sn_i)$$

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

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

$$\texttt{justify}\big(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}\big) \wedge \mathtt{C}_{js4} \Rightarrow \texttt{assert}(ag_j, ag_i, \overline{cq})$$

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

$$\texttt{justify}\big(ag_i, ag_j, \langle S, c\rangle^{\theta}_{sn_i}\big) \wedge \mathtt{C}_{js6} \Rightarrow \texttt{justify}\big(ag_j, ag_i, \langle S', c\rangle^{\theta}_{sn_j}\big)$$

where:

$$\mathtt{C}_{js1} = \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \notin \Delta_{ag_j}$$

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

$$\mathtt{C}_{js3} = \exists\langle S', \overline{p}\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', \overline{p}\rangle^{\theta}_{sn_j} \wedge \overline{p} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Op}(ag_j) \wedge \mathtt{subject}(p)$$

$$\mathtt{C}_{js4} = \exists\overline{cq}\theta : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \overline{cq}\theta \wedge \overline{cq}\theta \notin \mathtt{CS}_{ag_j} \wedge cq\theta \in \mathcal{CQ} \wedge \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{ag_j}$$

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

$$\mathtt{C}_{js6} = \exists\langle S', p\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', p\rangle^{\theta}_{sn_j} \wedge \langle S', p\rangle^{\theta}_{sn_j} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Pr}(ag_j) \wedge \mathtt{subject}(p)$$

When the agent is not aware of the argumentation schemes used by the other agent, it questions the other agent about such scheme, so that it will be able to properly evaluate that argument when receiving the information about the argumentation scheme used to instantiate it. Otherwise, the agent will accept the subject of the dialogue, $\mathtt{C}_{js2}$, if the justification received from the proponent has made the subject of the dialogue acceptable to it, 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, $\mathtt{C}_{js3}$, or it will assert that some critical question related to that argument are not positively answered, $\mathtt{C}_{js4}$. 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, $\mathtt{C}_{js5}$. In the final case, the agent sends a new argument[13] to support the subject of the dialogue, $\mathtt{C}_{js6}$.

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

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

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

where:

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

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.

Question Scheme Rule: The dialogue rule that restricts the moves when an agent receives a `question_scheme` message is:

$$\mathtt{question\_scheme}(ag_i, ag_j, sn_i) \wedge \mathtt{C}_{qsc1} \Rightarrow \mathtt{inform\_scheme}\big(ag_j, ag_i, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big)$$

where:

$$\mathtt{C}_{qsc1} = \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{ag_j}$$

An agent is supposed to be aware of the argumentation scheme previously used to instantiate an argument used in that particular dialogue move, thus it will provide such an argumentation scheme when questioned.

Inform Scheme Rules: The dialogue rules that restricts the moves when an agent receives a `question_scheme` message are:

$$\mathtt{inform\_scheme}\big(ag_i, ag_j, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big) \wedge \mathtt{C}_{iss1} \Rightarrow \mathtt{accept}(ag_j, ag_i, p)$$
$$\mathtt{inform\_scheme}\big(ag_i, ag_j, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big) \wedge \mathtt{C}_{iss2} \Rightarrow \mathtt{assert}(ag_j, ag_i, \overline{p})$$
$$\mathtt{inform\_scheme}\big(ag_i, ag_j, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big) \wedge \mathtt{C}_{iss3} \Rightarrow \mathtt{assert}(ag_j, ag_i, \overline{cq})$$
$$\mathtt{inform\_scheme}\big(ag_i, ag_j, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big) \wedge \mathtt{C}_{iss4} \Rightarrow \mathtt{closedialogue}(ag_j, ag_i)$$
$$\mathtt{inform\_scheme}\big(ag_i, ag_j, \langle sn_i, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big) \wedge \mathtt{C}_{iss5} \Rightarrow \mathtt{justify}\big(ag_j, ag_i, \mathtt{S}'\big)$$

where:

$$\mathtt{C}_{iss1} = \nexists \langle S', \overline{p}\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', \overline{p}\rangle^{\theta}_{sn_j} \wedge \mathtt{Op}(ag_j) \wedge \mathtt{subject}(p)$$
$$\mathtt{C}_{iss2} = \exists \langle S', \overline{p}\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', \overline{p}\rangle^{\theta}_{sn_j} \wedge \overline{p} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Op}(ag_j) \wedge \mathtt{subject}(p)$$
$$\mathtt{C}_{iss3} = \exists \overline{cq}\theta : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \overline{cq}\theta \wedge \overline{cq}\theta \notin \mathtt{CS}_{ag_j} \wedge cq\theta \in \mathcal{CQ}$$
$$\mathtt{C}_{iss4} = \nexists \langle S', p\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', p\rangle^{\theta}_{sn_j} \wedge \langle S', p\rangle^{\theta}_{sn_j} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Pr}(ag_j) \wedge \mathtt{subject}(p)$$
$$\mathtt{C}_{iss5} = \exists \langle S', p\rangle^{\theta}_{sn_j} : (\Delta_{ag_j} \cup \mathtt{CS}_{ag_i}) \models \langle S', p\rangle^{\theta}_{sn_j} \wedge \langle S', p\rangle^{\theta}_{sn_j} \notin \mathtt{CS}_{ag_j} \wedge \mathtt{Pr}(ag_j) \wedge \mathtt{subject}(p)$$

The dialogue then follows as when the agent receives a justify move being aware of the argumentation scheme used by ag_i, recovering the subject of the dialogue, accepting the subject of the dialogue, $\mathtt{C}_{iss1}$, if the justification received from the proponent has made the subject of the dialogue acceptable to it, 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, $\mathtt{C}_{iss2}$, or it will assert that some critical question related to that argument are not positively answered, $\mathtt{C}_{iss3}$. 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, $\mathtt{C}_{iss4}$. In the final case, the agent sends a new argument[14] to support the subject of the dialogue, $\mathtt{C}_{iss5}$.

5.3.2. Implementation in Jason agents

The dialogue rules presented in the previous section can be easily implemented in multi-agent platforms in which agent practical reasoning is inspired by reactive planning systems, such as the Jason platform [12]. In Jason, when an agent receives a message, an event of the form +!msg_received(Sender, Performative, Content) is generated to the receiver agent, and the agent can handle that event thereby generating new goals. To achieve their goals, agents look for plans. Thus, the dialogue rules presented in this section can be implemented as agents plans that aim to respond to received messages.

The number of plans to handle the event of receiving a particular message will be equivalent to the number of dialogue rules that restrict the next possible move with which the agent can respond. For example, the **assert rules** presented in the previous section can be implemented using the following agent plans:

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

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

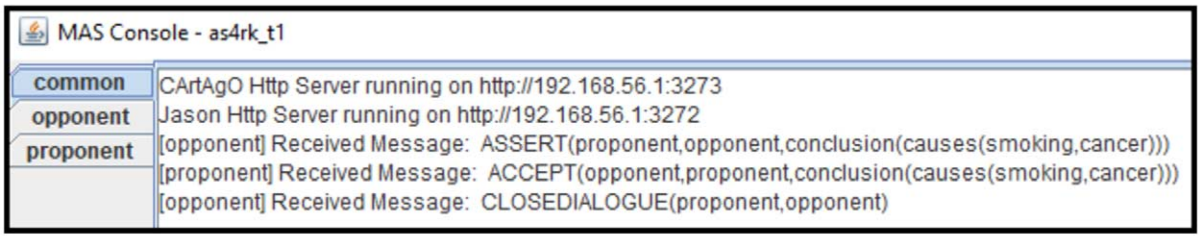
Similarly, other conditions used in dialogue rules can be easily implemented using our framework through a code library that we implemented and made publicly available [47,55].

5.3.3. Experiments

We ran some experiments to show different dialogues resulting from the protocol presented in Section 5.3 and implemented in our framework. In our experiments, two agents, named *proponent* and *opponent*, use the argumentation scheme *role to know* in order to argue about whether *smoking causes cancer* or not, based on assertions by hospital staff. So agent *proponent* starts a dialogue asserting that causes(smoking, cancer), which becomes the subject of the dialogue. The *proponent*'s knowledge is denoted by $\Delta_{proponent}$, as defined below.

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

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



```
CArtAgO Http Server running on http://192.168.56.1:3273
Jason Http Server running on http://192.168.56.1:3272
[opponent] Received Message:  ASSERT(proponent,opponent,conclusion(causes(smoking,cancer)))
[proponent] Received Message:  ACCEPT(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  CLOSEDIALOGUE(proponent,opponent)
```

Fig. 5. First example.

In the case where agent *opponent* has no argument against the subject of the dialogue, i.e., $\mathtt{causes(smoking, cancer)}$, it accepts the opening assertion, as shown in the output from our implementation in Fig. 5.

In the case where agent *proponent* has an argument against the subject of the dialogue, we have a different output.

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

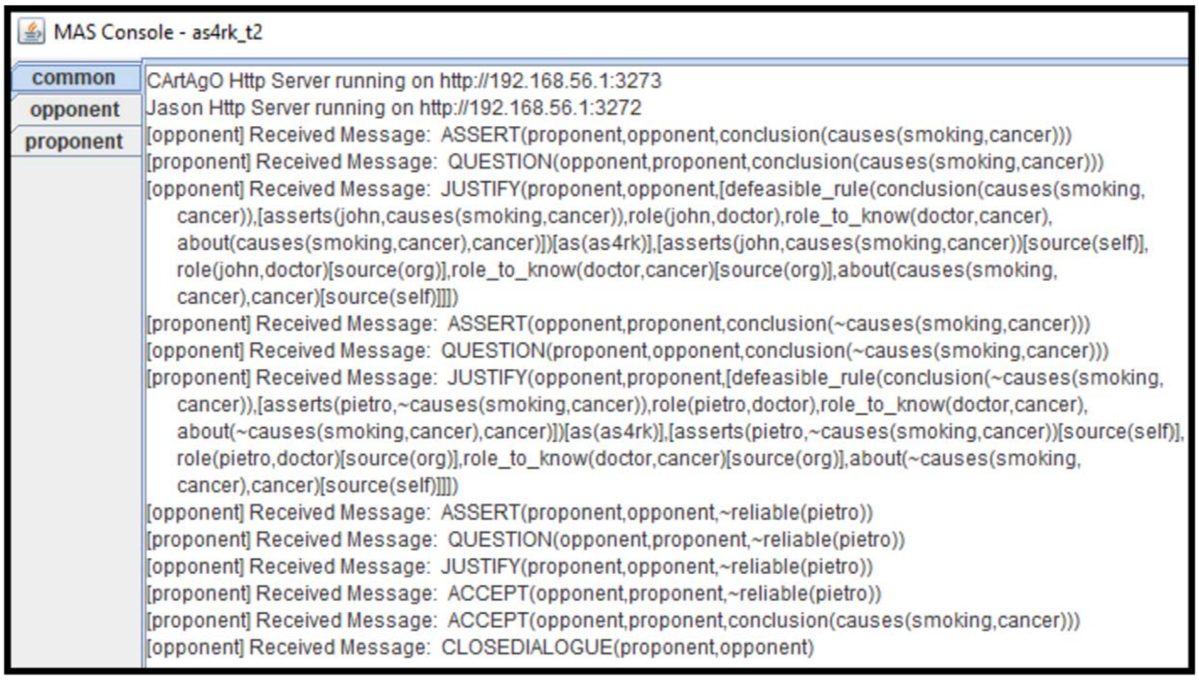
Considering the *opponent*'s knowledge represented as $\Delta_{opponent}$, in which the *opponent* agent assumes that *Pietro* is reliable and he is a doctor, initially the *opponent* has an argument against the subject of the dialogue. But, after receiving the information that Pietro is not reliable from *proponent*, it accepts the subject of the dialogue. The output of this dialogue is shown in Fig. 6.

These two first examples demonstrate dialogues that occur when we do not have chained arguments. Using chained arguments, agents may have more interesting dialogues. For example, continuing our example from Section 4.1, imagine that the *proponent* agent has asked reliability advisers about whether Pietro and John are reliable sources of information or not. Then, the *proponent* is able to provide an argument supporting that Pietro is not reliable, i.e., $\mathtt{\neg reliable(pietro)}$, based on those advisers' assertions (in our example, Ana's assertion). The output of this dialogue is showed in Fig. 7.

The dialogue shown in Fig. 7 demonstrates an example in which agents are able to argue about the critical questions pointed by the argumentation scheme used to instantiate the arguments used during the dialogues. Besides, agents are also able to argue about the premises of arguments. For example, imagine that the *opponent* agent believes that Pietro is a doctor because he is dressing white and he is in the hospital, i.e., $\mathtt{role(pietro, doctor)}$ is a belief the opponent creates based on its own perception; however, Pietro is actually a member of the cleaning staff. Also, imagine that the *proponent* knows that Pietro is a cleaner because the hospital director Mathew has told so. Thus, continuing our example, also based on the example from Section 3, the output of this dialogue is shown in Fig. 8.

6. Discussion

There are some interesting properties of our approach that are important for the development of multi-agent systems when applying our argumentation framework. One important property is that our approach is *general* regarding the computational representation of argumentation schemes of different forms, with

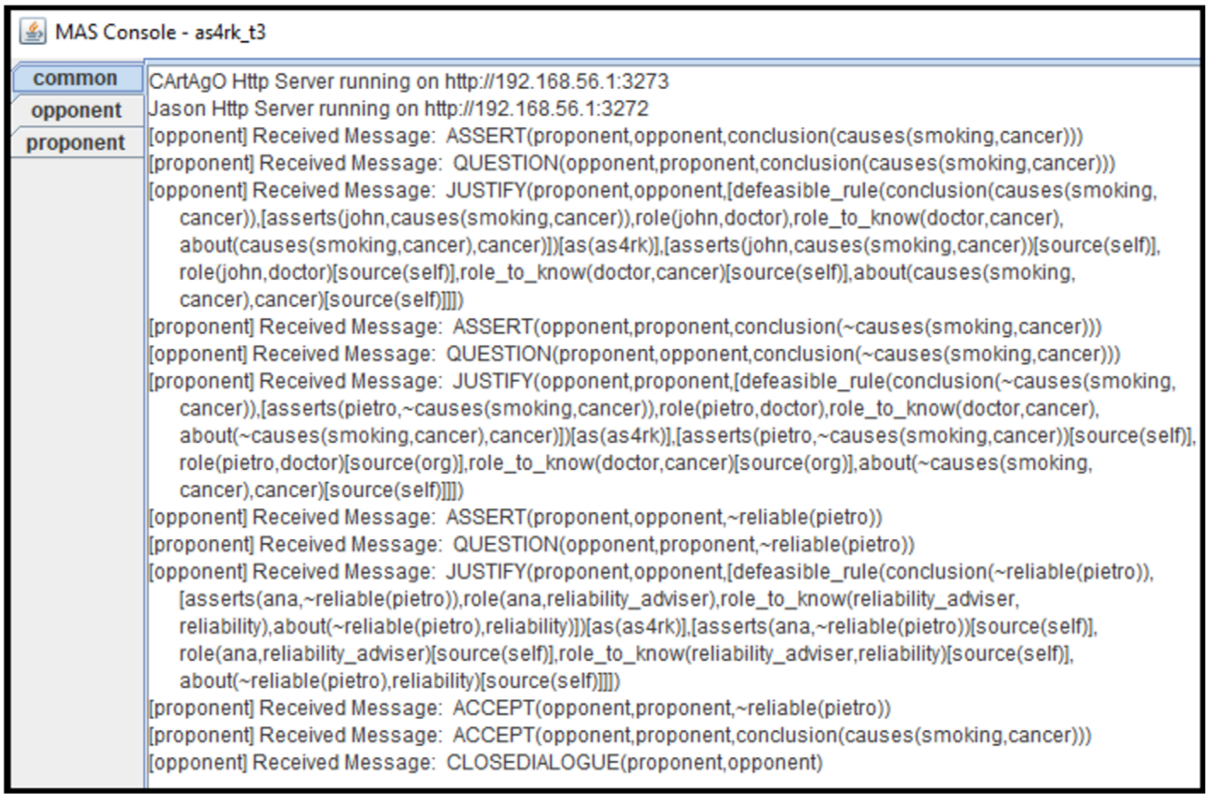


```
CArtAgO Http Server running on http://192.168.56.1:3273
Jason Http Server running on http://192.168.56.1:3272
[opponent] Received Message:  ASSERT(proponent,opponent,conclusion(causes(smoking,cancer)))
[proponent] Received Message:  QUESTION(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  JUSTIFY(proponent,opponent,[defeasible_rule(conclusion(causes(smoking,
    cancer)),[asserts(john,causes(smoking,cancer)),role(john,doctor),role_to_know(doctor,cancer),
    about(causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(john,causes(smoking,cancer))[source(self)],
    role(john,doctor)[source(org)],role_to_know(doctor,cancer)[source(org)],about(causes(smoking,
    cancer),cancer)[source(self)]]])
[proponent] Received Message:  ASSERT(opponent,proponent,conclusion(~causes(smoking,cancer)))
[opponent] Received Message:  QUESTION(proponent,opponent,conclusion(~causes(smoking,cancer)))
[proponent] Received Message:  JUSTIFY(opponent,proponent,[defeasible_rule(conclusion(~causes(smoking,
    cancer)),[asserts(pietro,~causes(smoking,cancer)),role(pietro,doctor),role_to_know(doctor,cancer),
    about(~causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(pietro,~causes(smoking,cancer))[source(self)],
    role(pietro,doctor)[source(org)],role_to_know(doctor,cancer)[source(org)],about(~causes(smoking,
    cancer),cancer)[source(self)]]])
[opponent] Received Message:  ASSERT(proponent,opponent,~reliable(pietro))
[proponent] Received Message:  QUESTION(opponent,proponent,~reliable(pietro))
[opponent] Received Message:  JUSTIFY(proponent,opponent,~reliable(pietro))
[proponent] Received Message:  ACCEPT(opponent,proponent,~reliable(pietro))
[proponent] Received Message:  ACCEPT(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  CLOSEDIALOGUE(proponent,opponent)
```

Fig. 6. Second example.

or without implicit critical questions, with any number of premises and critical questions, etc. The generality of the knowledge representation is further discussed in Panisson's PhD thesis [44], in which a set of argumentation schemes from the literature are represented using the same formalisation used in this paper. It is shown that our approach allows us to represent argumentation schemes of different levels of specificity, including those that use chained arguments (supporting premises and critical questions), as well as argumentation schemes with and without critical questions. To the best of our knowledge, there is no other computational representation for argumentation schemes with critical questions, whether formalised or implemented, in the specific context of agent-oriented programming languages.

Our approach keeps the essence of argumentation schemes discussed by Walton [89] into the knowledge representation, reasoning, and dialogues, while also keeping the knowledge engineering of argumentation schemes as straightforward as possible. To model an argumentation scheme into our approach, engineers only need to think about the premises, conclusion, and critical question that will implement that reasoning pattern, representing such information using first-order formulas and linking them by a defeasible inference rule (premises and conclusion) and labels (inference rule and associated critical questions). After that, in case there are critical questions or premises resulting from other argumentation schemes (if the application requires nested argumentation schemes), they only need to model those argumentation schemes in the same way, matching the conclusion of that argumentation scheme with the corresponding premise or critical question. When reasoning, agents are going to evaluate arguments that are instances of those argumentation schemes, and being aware of the reasoning pattern being used to instantiate a particular argument, agents are able to link the critical questions associated with it, bringing to light that implicit information used to construct the argument, so properly evaluating its validity. Most importantly, during dialogues, agents will communicate arguments similar to how humans do. Also, by identifying the reasoning pattern used by others when instantiating an argument, agents will have a

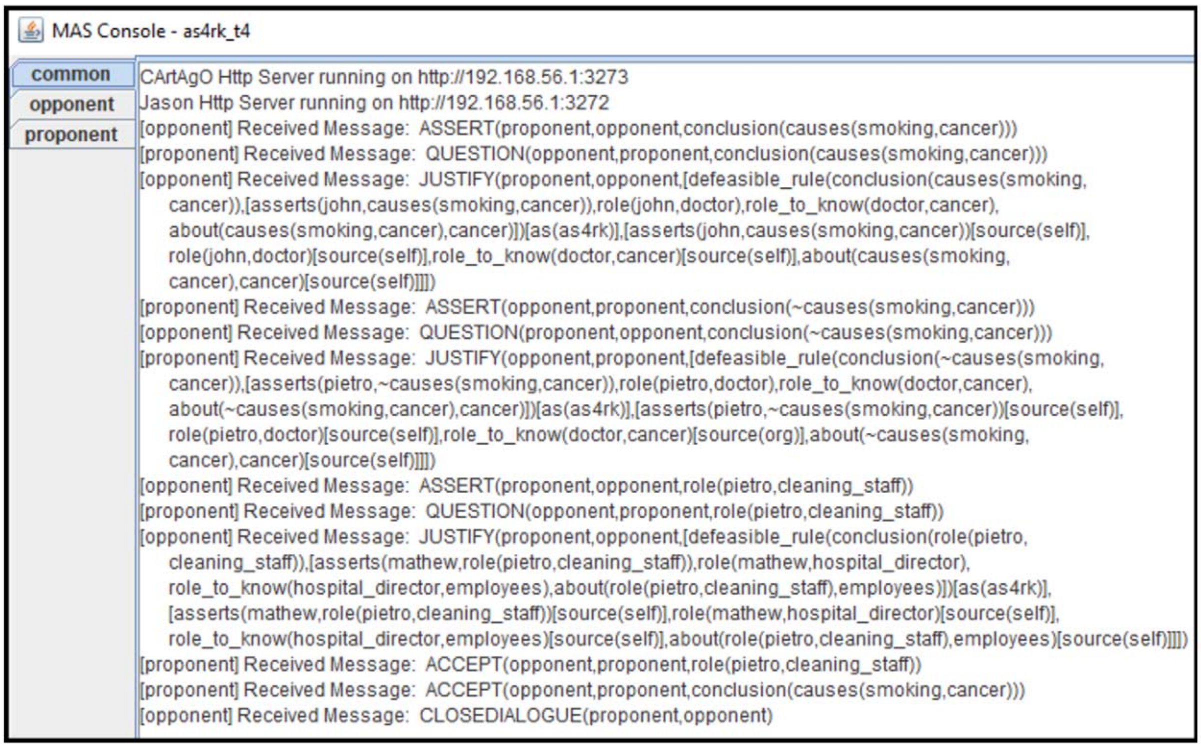


```
CArtAgO Http Server running on http://192.168.56.1:3273
Jason Http Server running on http://192.168.56.1:3272
[opponent] Received Message:  ASSERT(proponent,opponent,conclusion(causes(smoking,cancer)))
[proponent] Received Message:  QUESTION(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  JUSTIFY(proponent,opponent,[defeasible_rule(conclusion(causes(smoking,
    cancer)),[asserts(john,causes(smoking,cancer)),role(john,doctor),role_to_know(doctor,cancer),
    about(causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(john,causes(smoking,cancer))[source(self)],
    role(john,doctor)[source(self)],role_to_know(doctor,cancer)[source(self)],about(causes(smoking,
    cancer),cancer)[source(self)]]])
[proponent] Received Message:  ASSERT(opponent,proponent,conclusion(~causes(smoking,cancer)))
[opponent] Received Message:  QUESTION(proponent,opponent,conclusion(~causes(smoking,cancer)))
[proponent] Received Message:  JUSTIFY(opponent,proponent,[defeasible_rule(conclusion(~causes(smoking,
    cancer)),[asserts(pietro,~causes(smoking,cancer)),role(pietro,doctor),role_to_know(doctor,cancer),
    about(~causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(pietro,~causes(smoking,cancer))[source(self)],
    role(pietro,doctor)[source(org)],role_to_know(doctor,cancer)[source(org)],about(~causes(smoking,
    cancer),cancer)[source(self)]]])
[opponent] Received Message:  ASSERT(proponent,opponent,~reliable(pietro))
[proponent] Received Message:  QUESTION(opponent,proponent,~reliable(pietro))
[opponent] Received Message:  JUSTIFY(proponent,opponent,[defeasible_rule(conclusion(~reliable(pietro)),
    [asserts(ana,~reliable(pietro)),role(ana,reliability_adviser),role_to_know(reliability_adviser,
    reliability),about(~reliable(pietro),reliability)])[as(as4rk)],[asserts(ana,~reliable(pietro))[source(self)],
    role(ana,reliability_adviser)[source(self)],role_to_know(reliability_adviser,reliability)[source(self)],
    about(~reliable(pietro),reliability)[source(self)]]])
[proponent] Received Message:  ACCEPT(opponent,proponent,~reliable(pietro))
[proponent] Received Message:  ACCEPT(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  CLOSEDIALOGUE(proponent,opponent)
```

Fig. 7. Third example.

complete understanding of all information used by the speaker to evaluate the validity of the argument, including the implicit information pointed by the critical questions. It is very different from including those critical questions as additional premises into the arguments: in our approach they need not be communicated by agents, which reflects rationality and intelligence as observed in human dialogues, also reflecting some of Grice's maxims [29]. Also, it is fundamentally different from modelling critical questions as undercutting arguments, in which the receiver of an argument only will be able to evaluate those critical questions when aware of such arguments (which would not be provided by a speaker, typically). Providing the argumentation scheme used to instantiate an argument would be natural for many types of dialogues and in various scenarios such as legal argumentation, for example.

Our approach takes into consideration the role of critical questions in argumentation schemes, covering all the possible roles of critical questions pointed out in [84], i.e., critical questions that: (i) criticise premises, as shown in our examples when criticising the role of an agent; (ii) point out exceptional situations in which the scheme could not be used, as shown in our examples when criticising the reliability of an agent; and (iii) representing conditions for the schemes' use, given that we are able to model context-dependent argumentation schemes including critical questions of the type: "Is C (the current context) the right context?", representing it as $\texttt{context(C)}_{[sn]}$, that is, requiring a particular context `C` that should hold when attempting to instantiate arguments from that scheme, according to Definition 3. Note that such context information is not part of the argument itself, but it is required to reach an acceptable (and rational) conclusion from that argumentation scheme.



```
CArtAgO Http Server running on http://192.168.56.1:3273
Jason Http Server running on http://192.168.56.1:3272
[opponent] Received Message:  ASSERT(proponent,opponent,conclusion(causes(smoking,cancer)))
[proponent] Received Message:  QUESTION(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  JUSTIFY(proponent,opponent,[defeasible_rule(conclusion(causes(smoking,
    cancer)),[asserts(john,causes(smoking,cancer)),role(john,doctor),role_to_know(doctor,cancer),
    about(causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(john,causes(smoking,cancer))[source(self)],
    role(john,doctor)[source(self)],role_to_know(doctor,cancer)[source(self)],about(causes(smoking,
    cancer),cancer)[source(self)]]])
[proponent] Received Message:  ASSERT(opponent,proponent,conclusion(~causes(smoking,cancer)))
[opponent] Received Message:  QUESTION(proponent,opponent,conclusion(~causes(smoking,cancer)))
[proponent] Received Message:  JUSTIFY(opponent,proponent,[defeasible_rule(conclusion(~causes(smoking,
    cancer)),[asserts(pietro,~causes(smoking,cancer)),role(pietro,doctor),role_to_know(doctor,cancer),
    about(~causes(smoking,cancer),cancer)])[as(as4rk)],[asserts(pietro,~causes(smoking,cancer))[source(self)],
    role(pietro,doctor)[source(self)],role_to_know(doctor,cancer)[source(org)],about(~causes(smoking,
    cancer),cancer)[source(self)]]])
[opponent] Received Message:  ASSERT(proponent,opponent,role(pietro,cleaning_staff))
[proponent] Received Message:  QUESTION(opponent,proponent,role(pietro,cleaning_staff))
[opponent] Received Message:  JUSTIFY(proponent,opponent,[defeasible_rule(conclusion(role(pietro,
    cleaning_staff)),[asserts(mathew,role(pietro,cleaning_staff)),role(mathew,hospital_director),
    role_to_know(hospital_director,employees),about(role(pietro,cleaning_staff),employees)])[as(as4rk)],
    [asserts(mathew,role(pietro,cleaning_staff))[source(self)],role(mathew,hospital_director)[source(self)],
    role_to_know(hospital_director,employees)[source(self)],about(role(pietro,cleaning_staff),employees)[source(self)]]])
[proponent] Received Message:  ACCEPT(opponent,proponent,role(pietro,cleaning_staff))
[proponent] Received Message:  ACCEPT(opponent,proponent,conclusion(causes(smoking,cancer)))
[opponent] Received Message:  CLOSEDIALOGUE(proponent,opponent)
```

Fig. 8. Fourth example.

Using this computational representation for argumentation schemes combined with the consideration of all roles of the critical questions, we provide a refined and modular argumentation-based reasoning mechanism, in which agents first instantiate and evaluate arguments individually, according to Definition 3, using those acceptable instances of argumentation schemes to check which ones are collectively acceptable given a particular argumentation semantics, according to Definition 6. To the best of our knowledge, there is no other work that proposed argumentation-based reasoning using a general structure for argumentation schemes with implicit critical questions. Also, we provided an empirical evaluation, showing how our (open source) implementation performs according to different levels of specificity for argumentation schemes (single and chained argumentation schemes, with varying numbers of critical questions and premises). Those results might guide the knowledge engineering process of modelling argumentation schemes according to the application needs.

Further, our framework has important properties regarding argumentation-based dialogues: (i) first, we allow agents to be able to communicate argumentation schemes when necessary, solving the problem of scheme awareness as discussed by Wells [91]. Thus, when an agent receives an argument and it is not able to identify which argumentation scheme has been used to instantiate that argument (and consequently not being able to evaluate individually that argument), the agent can ask other agents to inform which argumentation schemes have been used; (ii) second, when engineering multi-agent systems, our approach may guide the modelling of argumentation-based protocols, in regards to the purpose of such dialogues and supporting agents with different levels of bounded rationality; (iii) third, when considering (deep) disagreements among the participants, our approach allows agents to engage in subdialogues concerning the critical questions related to the argumentation schemes used to instantiate arguments, in the fashion of a dialogue-subdialogue structure. Althgouh this is part of our ongoing research efforts, note that it means agents possibly entering into a subdialogue about some implicit information used in that argument, as shown in the dialogue in Fig. 7, which has feature that may be regarded as intelligent behaviour for autonomous agents.

Finally, our work provides interesting directions for the development of Explainable AI [30,31] and the interaction between humans and agents in the context of Hybrid Intelligence [1]. When modelling argumentation schemes to this end (explainability), agents are directly able to create, evaluate, and communicate arguments providing explanations for one another or for humans [41]. Templates of the argumentation schemes can be used to provide a translation between the natural language and the computational representation of arguments, allowing for more sophisticated human-agent interactions [51], or even using chatbot technologies [20–22]. These research directions are part of our ongoing work.

7. Related work

Related work exists using argumentation schemes in multi-agent systems [28,34,46,57,79,81], all of which concern the modelling and using one (two in [46]) argumentation scheme that covers the application needs. However, none of those approaches proposes a general representation of argumentation schemes in agent-oriented programming languages.

In [71], the authors have described a formal account for argumentation schemes; however, such formal representation does not directly give to agents a mechanism either for reasoning with schemes or for constructing arguments using schemes. That work focused on the integration of the ARAUCARIA [70] tool (used to specify argumentation schemes) and agent platforms. Although we took inspiration from [71], we proposed a general structure for argumentation schemes in agent-oriented programming languages, which directly enables agents to use such representation for reasoning and communication.

In [84], the authors described a method to investigate argumentation schemes, which consists of: (i) determining the relevant types of sentences; (ii) determining the argumentation schemes; (iii) determining the exceptions blocking the use of the argumentation schemes; (iv) determining the conditions for the use of the argumentation scheme. We took inspiration from [84], using the desiderata for computational models of argumentation schemes, and [89], as a guide to propose a general structure for argumentation schemes, also considering all roles that critical questions could play.

In [15], the authors introduce a proposal to represent and share arguments, called the Argument Interchange Format (AIF), which represents a standard "abstract model" established by researchers across the fields of argumentation, artificial intelligence, and multi-agent systems. As described in [15], one of the major barriers to the development and practical deployment of an argumentation system is the lack of shared, agreed notation for argumentation and arguments. AIF aims to establish the following principles: (i) a machine-readable syntax for argumentation schemes; (ii) an explicit and machine-processable semantics for argumentation schemes; (iii) a unified abstract model with multiple reifications; (iv) core concepts within multiple extensions. AIF has at its core arguments and argument networks, communication (locutions and protocols), and context (the environment in which argumentation takes place). The AIF has been extended to capture dialogic argumentation in [72]. Furthermore, in [66], the authors also extend AIF to incorporate the representation of argumentation schemes by Walton [89]. Thus, critical questions are considered through a particular ontological relation named *hasException*, which points out the exceptions that can lead an argument to being considered unacceptable by a reasoning agent. Also, an ontological relation named *hasPresumption* is used to represent the information that is not explicit in the argument, as in the case of the credibility of a source in the argumentation scheme from role to know described in Section 2.2. AIF also has been used in interesting proposals that aim to create a Web infrastructure (Argument Web) that allows for storage and automatic retrieval and analysis of linked argument data [15]. Although the AIF provides a means for sharing arguments, to the best of our knowledge, there is no work on the integration of AIF and agent-oriented programming languages, and such an integration would be an interesting approach to pursue. In our approach, the AIF could be used to represent arguments at a higher level, allowing agents to share that knowledge in a multi-agent system. Those arguments and argumentation schemes could be translated into an agent-oriented programming language based on our approach, using our knowledge representation, which can be directly manipulated by agents, considering that agents can manipulate arguments and argumentation schemes represented using our approach in the same manner as they manipulate their beliefs.

In [93], the authors classify different levels of representation for argumentation schemes found in the literature, namely: (i) atomic level; (ii) functional roles and typed propositional functions; (iii) functional roles and instantiated predicates; (iv) labelled roles and strings; and (v) canonical sentences. Thus, the authors in [93] propose a functional language for computational analysis of argumentation schemes, so that different argumentation schemes can be compared. Although our aim is not to investigate the relationship between different argumentation schemes but to represent argumentation schemes in agent-oriented programming languages in order for the agents to be directly able to manipulate them, we took some inspiration from [93], inheriting the most refined and detailed representation for argumentation schemes given by (ii).

In [91], the authors describe how argumentation schemes could be exploited in dialogue games, introducing the idea of "scheme awareness", providing not only a guideline for the development of new dialogue games using argumentation schemes but also a mechanism to extend dialogue-game frameworks to account for scheme awareness. They claim that current approaches to dialogue games can be categorised into three levels: (i) games unable to utilise argumentation schemes; (ii) games able to utilise a single scheme; and (iii) games able to utilise multiple/arbitrary schemes. Also, they claim that currently there are no approaches to games at level (iii). Whilst our work focuses on how agents represent, manipulate, and instantiate arguments from an internal representation of argumentation schemes to carry out argumentation-based reasoning, our approach forms the basis for argumentation-based dialogues using multiple argumentation schemes, i.e., as in (iii), given that agents can reason over any number of argumentation schemes, communicate those arguments, and communicate the argumentation schemes used to instantiate those arguments when necessary.

In [33], the authors present an approach for structured argumentation in multi-agent system dialogues. The authors claim that the use of argumentation in inter-agent dialogues may be beneficial to the agents. Also, they state that existing work on the experimental evaluation of the benefits of argumentation in agent dialogues make use of very simple models of argumentation, in which arguments have no or very little structure. We follow some directions pointed out in [33], defining a complete structure for arguments based on argumentation schemes, and experimentally evaluating our approach for argumentation-based dialogues.

The idea of labelling arguments with meta-information used in this paper comes from Gabbay's work [24], further developed by other authors in [13,14,16,45,56]. While all that work focuses on modelling an argumentation framework in which the labels play an important role in the inference mechanism (mostly propagating strength of premises to the conclusions of arguments), here we propose a more modest use in which labels are used to make reference to argumentation schemes used to instantiate those arguments, pointing out the critical questions related to those instances of arguments.

In [41], the authors explore two important contributions of Walton's work to AI (particularly in the design of autonomous software agents able to reason and argue with one another), namely argumentation schemes and dialogue protocols, describing how they may apply to current research on Explainable AI by automated decision-making systems. The authors also mention an example of how argumentation schemes could be chained, not only regarding their premises and conclusion but also regarding the critical questions, in which the answer of a critical question could be the conclusion of an argumentation scheme. The work presented by [41] inspired us to develop our argumentation-scheme-centred framework, keeping the essence of argumentation schemes presented by Walton's work. Our approach moves towards the directions pointed out in [41] for using argumentation schemes in multi-agent systems, which also can be used to support explainability. Some initial steps towards this direction are found in [51].

In [6,64], the authors present a formal account of legal reasoning, which can be seen as moving from evidence to facts, from facts to factors, and from factors to legal consequences. Their previous work, the CATO system presented in [2] and [8], implemented a single step of argumentation, concerning the last phase for legal reasoning, i.e., moving from factors to legal cases. Thus, they extend CATO system to bring the assignment of factors within the scope of the system, i.e., moving from facts to factors, and so open this aspect to explicit argumentation [6]. The authors propose CATO style argumentation schemes, in which schemes and undercutting attacks associated with them are formalised as defeasible inference rules within the ASPIC+ framework [42]. Differently from our approach, they use CATO style argumentation schemes. Argumentation schemes correspond to defeasible inference rules in ASPIC+, in which premises can be axioms (indisputable facts) or the conclusion from another argumentation scheme. Attacks are captured by defining undercutting CATO style argumentation schemes, which allow instantiating arguments against the application of another CATO style argumentation scheme. Undercutting arguments also may have undercutting arguments against their application. The work in [6,64] moves towards a similar direction, which considers first using argumentation schemes to instantiate arguments (for and against deciding for a plaintiff, in their case), rebutting each other, and later solving those conflicts using argument graphs (with preference, in their case). Their formalism allows using nested argumentation schemes, similar to our approach. However, while undercutting schemes enable capturing the critical questions related to an argumentation scheme, their approach requires using nested argumentation schemes to model critical questions. That is, undercutting schemes allow agents to look for arguments concluding that the previous scheme does not apply in that particular case, implementing one particular role for critical questions [84]. Also, the approach in [64] may require more effort from the knowledge engineering point of view, in which modelling critical questions requires additional argumentation schemes.

8. Conclusion

In this paper, we have presented an argumentation framework developed on top of an agent-oriented programming language, in which argumentation schemes are at the core of the framework.

We first proposed a general structure for argumentation schemes represented in agent-oriented programming languages, in which argumentation schemes of different levels of specificity can be modelled, allowing the development of applications that use argumentation schemes with and without implicit critical questions, as well as using single or chained argumentation schemes.

We then presented an argumentation-based reasoning mechanism, in which agents consider those modelled argumentation schemes in order to construct and define the acceptability of arguments. In order to evaluate our framework in regards to argumentation-based reasoning, we implemented our framework and ran some experiments to show the performance of agents that follow our approach, considering different levels of specificity of argumentation schemes, i.e., using single and chained argumentation schemes. Our results confirm our initial hypothesis that using argumentation schemes without chained arguments is computationally more efficient than using chained arguments. These results reflect on the development of intelligent agents with different levels of bounded rationality, and also might guide knowledge engineers in the process of modelling argumentation schemes. While using a single argumentation scheme has been sufficient for the development of many multi-agent applications using argumentation [28,34,57,79,81], there are application domains in which a more complex reasoning process is required, for example, [46,64], and our implementation supports both approaches.

We also presented an approach for argumentation-based dialogues using argumentation schemes, defining a protocol for argumentation-based dialogues that combines deliberation and information-seeking dialogues. We implemented that protocol for Jason agents [12], and showed different lines of argumentation using that protocol. Not only did we validate our framework through those examples, but we also provided a solution for the problem of scheme awareness described in [91].

Finally, we discussed how our work contributes to the development of the area of argumentation in multi-agent systems. We also discuss how our work moves towards the practical use of argumentation as a technique for explainable AI [30,31]. In future work, we intend to implement applications with human-agent interaction using argumentation schemes. We intend to use argumentation schemes represented in both natural language and the formal representation presented in this work, as in [51], so that agents can use the appropriate representation depending on whether they are communicating with software agents or with human beings.

Acknowledgements

The authors dedicate this work to the memory of Professor Douglas Walton. Alison had the pleasure to meet Professor Douglas Walton in person during COMMA 2018. During one of Alison's presentations during that conference, Professor Walton asked a question, concluding his speech with a compliment, saying that our approach to knowledge representation for argumentation schemes was *"elegant"*. That was an unforgettable moment for Alison, considering Alison's great admiration of Walton's work and as a person. Also, it was that comment (and further discussion during that conference) that motivated us to pursue many of the directions taken in this work. Rafael Bordini and Alison Panisson gratefully acknowledge partial funding from CNPq and CAPES.

Appendix A. Semantics for speech-acts using argumentation schemes

In this appendix, we formalise the operational semantics for the speech acts described above. We define the semantics of speech acts for argumentation-based dialogues in AgentSpeak [68] (and in particular the Jason dialect [12]) using a widely-known method for giving operational semantics to programming languages [61]. Although we use AgentSpeak, the formal semantics makes reference for general components of the BDI architecture, therefore any language based on concepts such as beliefs, intentions, etc., could also benefit from our formalisation.

The operational semantics is given by a set of inference rules that define a transition relation between agent configurations $\langle ag, C, M, T, s\rangle$, and we here use the semantics for AgentSpeak originally defined in [85], where:[15]

- An agent ag is a set of beliefs bs and a set of plans ps.
- An agent circumstance C is a tuple $\langle I, E\rangle$ where:
  - I is a set of *intentions* $\{i, i', \dots\}$. Each intention i is a stack of partially instantiated plans.
  - E is a set of *events* $\{(te, i), (te', i'), \dots\}$. Each event is a pair (te, i), where te is a triggering event and i is an intention – a stack of plans in the case of an internal event, or the empty intention T in the 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 additions or deletions of goals generated by plans currently executing.
- M is a tuple $\langle In, Out\rangle$ whose components characterise the following aspects of communicating agents (note that communication is typically asynchronous):
  - In is the mail inbox: the MAS 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 atomic formulæ (beliefs), an argument, etc.
  - 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.
- 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 with such temporary information originally defined in [85]; in this paper, we only need the ι component, which keeps track of a particular intention being considered along the execution of a reasoning cycle.
- The current step within an agent's reasoning cycle is symbolically annotated by $s \in$ {ProcMsg, SelEv, RelPl, ApplPl, SelAppl, AddIM, SelInt, ExecInt, ClrInt}. Here, we make use of only ProcMsg, the step for processing a message from the agent's mail inbox, and ExecInt for executing the selected intention.
- 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 function, which is used to select one message from an agent's mail inbox, as originally defined in [85].

In the interest of readability, we adopt the following notational conventions in our semantic 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 in our semantics.
- We write: (i) $b[s(id)]$ to identify the origin of a belief b, where id is an agent identifier (s is an abbreviation for *source*).
- We use a function `prem(S)` which returns all premises in the support of the argument $\langle S, c\rangle$.

[15] We use here only the components that are needed to define our semantics.

A.1. Semantics for sending messages

(**ExActSndAssert**)

$$\frac{T_\iota = i[head \leftarrow .\texttt{send}(id, \texttt{assert}, c);h]}{\langle ag, C, M, T, \textsf{ExecInt}\rangle \longrightarrow \langle ag, C', M', T, \textsf{ProcMsg}\rangle}$$

where:

$$M'_{Out} = M_{Out} \cup \{\langle mid, id, \texttt{assert}, c\rangle\}$$
$$C'_I = (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$$

Sending an `assert`**,** `question`**,** `accept`**,** `question_scheme`**, and** `refuse` **message:** when an agent executes the internal action for sending a message with these performatives, that message is posted in the agent mailbox, M_{Out}, and the current agent intention is updated, removing the internal action, given that its execution is completed. In the semantic rule **ExActSndAssert**, the intention being considered is given by T_ι, and it corresponds to $i[head \leftarrow .\texttt{send}(id, \texttt{assert}, c);h]$, in which the current step of the plan adopted to reach that particular goal is to execute the action $.\texttt{send}(id, \texttt{assert}, c)$. Thus, after executing that action, that particular intention is updated to $i[head \leftarrow h]$, given that action has already been executed by the agent. Note that the semantic rules for the performatives `question`, `accept`, and `refuse` are similar to rule **ExActSndAssert**, thus we omit them in this paper.

(**ExActSndJustify**)

$$\frac{\begin{array}{c} T_\iota = i[head \leftarrow .\texttt{send}(id, \texttt{justify}, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}});h] \\ S \subset \Delta_{\texttt{ag}} \qquad \Delta_{\texttt{ag}} \models c \qquad \langle \texttt{sn}_\texttt{i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{ag} \qquad \forall p \in \mathcal{P},\ p\theta \in \texttt{prem}(S) \qquad c = \mathcal{C}\theta \end{array}}{\langle ag, C, M, T, \textsf{ExecInt}\rangle \longrightarrow \langle ag, C', M', T, \textsf{ProcMsg}\rangle}$$

where:

$$M'_{Out} = M_{Out} \cup \{\langle mid, id, \texttt{justify}, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}}\rangle\}$$
$$C'_I = (C_I \setminus \{T_\iota\}) \cup \{i[head \leftarrow h]\}$$

Sending a `justify` **message:** when an agent executes the internal action for sending a message with the performative `justify`, the agent needs to have an argument for that particular conclusion[16] that was drawn using the argumentation scheme $\texttt{sn}_\texttt{i}$, according to Definition 6. The corresponding message is posted in the agent's mailbox, M_{Out}, and the current agent intention is updated, removing the internal action, given that its execution is completed.

A.2. Semantics for receiving messages

(**Question**)

$$\frac{S_M(M_{In}) = \langle mid, sid, \texttt{question}, p\rangle}{\langle ag, C, M, T, \textsf{ProcMsg}\rangle \longrightarrow \langle ag, C', M', T, \textsf{ExecInt}\rangle}$$

[16]While we use, in this work, the *thoughtful* attitude [58,60], other agent attitudes could be used just as well.

where:

$$M'_{In} = M_{In} \setminus \big\{\langle mid, sid, \texttt{question}, p\rangle\big\}$$
$$C'_E = C_E \cup \big\{\big\langle +questions(sid, p), \mathsf{T}\big\rangle\big\}$$

Receiving `question`**,** `refuse`**,** `accept`**,** `assert` **messages:** the agent receiving the message will just remove the message from its mailbox and an event will be generated for that. The event is handled as usual by the agent's plans, which determine its strategy in the dialogue.

(QUESTION_SCHEME)

$$\frac{\begin{array}{c} S_M(M_{In}) = \langle mid, sid, \texttt{question\_scheme}, \texttt{sn}_\texttt{i}\rangle \\ \langle \texttt{sn}_\texttt{i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{\texttt{ag}} \end{array}}{\langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow \langle ag, C, M', T, \mathsf{ExecInt}\rangle}$$

where:

$$M'_{In} = M_{In} \setminus \big\{\langle mid, sid, \texttt{question\_scheme}, \texttt{sn}_\texttt{i}\rangle\big\}$$
$$M'_{Out} = M_{Out} \cup \big\{\big\langle mid, sid, \texttt{inform\_scheme}, \langle \texttt{sn}_\texttt{i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle\big\rangle\big\}$$

Receiving an `inform_scheme` **message:** the agent receiving the message will response the argumentation scheme questioned using a message with the performative `inform_scheme`, the agent needs to be aware of the argumentation scheme $\langle \texttt{sn}_\texttt{i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle$, according to Definition 7. The corresponding message is posted in the agent's mailbox, M_{Out}, and the current agent intention is updated, removing the internal action, given that its execution is completed.

(JUSTIFY)

$$\frac{\begin{array}{c} S_M(M_{In}) = \langle mid, sid, \texttt{justify}, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}}\rangle \\ \langle \texttt{sn}_\texttt{i}, \mathcal{C}, \mathcal{P}, \mathcal{CQ}\rangle \in \Delta_{\texttt{AS}} \qquad \forall p\theta \in S, p \in \mathcal{P} \qquad c = \mathcal{C}\theta \end{array}}{\langle ag, C, M, T, \mathsf{ProcMsg}\rangle \longrightarrow \langle ag', C', M', T, \mathsf{ExecInt}\rangle}$$

where:

$$M'_{In} = M_{In} \setminus \big\{\big\langle mid, sid, \texttt{justify}, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}}\big\rangle\big\}$$
$$ag'_{bs} = ag_{bs} \cup \{p\theta\big[s(sid)\big]| \quad \text{for all } p\theta \in S$$
$$C'_E = C_E \cup \big\{\big\langle +justifies\big(sid, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}}\big), \mathsf{T}\big\rangle\big\}$$

Receiving a `justify` **message:** when an agent selects a `justify` message from its mailbox, the message is removed from the mailbox, the agent's belief base is updated with all information contained in the support of the argument, annotating that formulæ have been received from the sender *sid*, and an event $+justifies(sid, \langle S, c\rangle^{\theta}_{\texttt{sn}_\texttt{i}})$ is generated for that. The event can be handled by the agent's plans according to its strategy in the dialogue.

References

[1] Z. Akata, D. Balliet, M. de Rijke, F. Dignum, V. Dignum, G. Eiben, A. Fokkens, D. Grossi, K. Hindriks, H. Hoos et al., A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, *Adaptive, Responsible, and Explainable Artificial Intelligence, Computer* **53**(8) (2020), 18–28.

[2] V.A. Aleven, Teaching case-based argumentation through a model and examples, Citeseer, 1997.

[3] L. Amgoud, N. Maudet and S. Parsons, Modeling dialogues using argumentation, in: *ICMAS*, IEEE Computer Society, 2000, pp. 31–38.

[4] L. Amgoud and S. Vesic, A formal analysis of the role of argumentation in negotiation dialogues, *J. Log. and Comput.* **22**(5) (2012), 957–978. doi:10.1093/logcom/exr037.

[5] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. Simari, M. Thimm and S. Villata, Towards artificial argumentation, *AI Magazine* **38**(3) (2017), 25–36. doi:10.1609/aimag.v38i3.2704.

[6] K. Atkinson, T.B.-C.H. Prakken and A. Wyner, Argumentation schemes for reasoning about factors with dimensions, in: *Legal Knowledge and Information Systems: JURIX 2013: The Twenty-Sixth Annual Conference*, Vol. 259, IOS Press, 2013, p. 39.

[7] T. Bench-Capon and K. Atkinson, Argumentation schemes: From informal logic to computational models, in: *Dialectics, Dialogue and Argumentation: An Examination of Douglas Walton's Theories of Reasoning and Argument*, 2010, pp. 103–114.

[8] T. Bench-Capon and G. Sartor, A model of legal reasoning with cases incorporating theories and values, *Artificial Intelligence* **150**(1–2) (2003), 97–143.

[9] J. Bentahar, R. Alam and Z. Maamar, An argumentation-based protocol for conflict resolution, in: *KR2008-Workshop on Knowledge Representation for Agents and MultiAgent Systems (KRAMAS 2008)*, 2008.

[10] P. Besnard, A. Garcia, A. Hunter, S. Modgil, H. Prakken, G. Simari and F. Toni, Introduction to structured argumentation, *Argument & Computation* **5**(1) (2014), 1–4. doi:10.1080/19462166.2013.869764.

[11] P. Besnard and A. Hunter, Constructing argument graphs with deductive arguments: A tutorial, *Argument & Computation* **5**(1) (2014), 5–30. doi:10.1080/19462166.2013.869765.

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

[13] M.C. Budán, M.G. Lucero, I. Viglizzo and G.R. Simari, A labeled argumentation framework, *Journal of Applied Logic* **13**(4) (2015), 534–553. doi:10.1016/j.jal.2015.02.005.

[14] M.C. Budán, G.I. Simari, I. Viglizzo and G.R. Simari, An approach to characterize graded entailment of arguments through a label-based framework, *International Journal of Approximate Reasoning* **82** (2017), 242–269. doi:10.1016/j.ijar.2016.12.016.

[15] C. Chesnevar, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, S. Willmott et al., Towards an argument interchange format, *The Knowledge Engineering Review* **21**(4) (2006), 293–316. doi:10.1017/S0269888906001044.

[16] C.I. Chesnevar and G.R. Simari, Modelling inference in argumentation through labelled deduction: Formalization and logical properties, *Logica Universalis* **1**(1) (2007), 93–124. doi:10.1007/s11787-006-0005-4.

[17] V. de Oliveira Gabriel, A.R. Panisson, R.H. Bordini, D.F. Adamatti and C.Z. Billa, Reasoning in BDI agents using Toulmin's argumentation model, *Theoretical Computer Science* **805** (2020), 76–91. doi:10.1016/j.tcs.2019.10.026.

[18] F. Dignum, B. Dunin-Keplicz and R. Verbrugge, Creating collective intention through dialogue, *Logic Journal of IGPL* **9**(2) (2001), 289–304. doi:10.1093/jigpal/9.2.289.

[19] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, *Artificial Intelligence* **77** (1995), 321–357. doi:10.1016/0004-3702(94)00041-X.

[20] D. Engelmann, J. Damasio, T. Krausburg, O. Borges, L.D. Cezar, A.R. Panisson and R.H. Bordini, Dial4JaCa – a demonstration, in: *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2021, pp. 346–350.

[21] D. Engelmann, J. Damasio, T. Krausburg, O. Borges, M. Colissi, A.R. Panisson and R.H. Bordini, Dial4JaCa – a communication interface between multi-agent systems and chatbots, in: *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2021, pp. 77–88.

[22] D.C. Engelmann, L.D. Cezar, A.R. Panisson and R.H. Bordini, A conversational agent to support hospital bed allocation, in: *Brazilian Conference on Intelligent Systems, BRACIS*, 2021.

[23] A. Freitas, D. Schmidt, A. Panisson, R.H. Bordini, F. Meneguzzi and R. Vieira, Applying ontologies and agent technologies to generate ambient intelligence applications, in: *Agent Technology for Intelligent Mobile Services and Smart Societies*, Springer, 2014, pp. 22–33.

[24] D.M. Gabbay, *Labelled Deductive Systems*, 1996.

[25] D.M. Gabbay, Introduction to labelled deductive systems, in: *Handbook of Philosophical Logic*, Springer, 2014, pp. 179–266. doi:10.1007/978-94-007-6600-6_3.

[26] A.J. García and G.R. Simari, Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers, *Argument & Computation* **5**(1) (2014), 63–88. doi:10.1080/19462166.2013.869767.

[27] A.J. García and G.R. Simari, Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers, *Argument & Computation* **5**(1) (2014), 63–88. doi:10.1080/19462166.2013.869767.

[28] N. Green, Implementing argumentation schemes as logic programs, in: *The 16th Workshop on Computational Models of Natural Argument*, CEUR, Vol. 30, 2016.

[29] H.P. Grice, Logic and conversation, in: *Speech Acts*, P. Cole and J. Morgan, eds, Academic Press, New York, 1975, pp. 41–58.

[30] D. Gunning, Explainable artificial intelligence (xai), *Defense Advanced Research Projects Agency (DARPA), nd Web* **2** (2017), 2.

[31] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf and G.-Z. Yang, XAI – explainable artificial intelligence, *Science Robotics* **4**(37) (2019). doi:10.1126/scirobotics.aay7120.

[32] E. Karafili, A.C. Kakas, N.I. Spanoudakis and E.C. Lupu, Argumentation-based security for social good, in: *AAAI*, 2017.

[33] E.M. Kok, J.-J.C. Meyer, H. Prakken and G.A. Vreeswijk, Testing the benfits of structured argumentation in multi-agent deliberation dialogues, in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 1411–1412.

[34] N. Kokciyan, I. Sassoon, A.P. Young, M. Chapman, T. Porat, M. Ashworth, V. Curcin, S. Modgil, S. Parsons and E. Sklar, Towards an argumentation system for supporting patients in self-managing their chronic conditions, in: *AAAI Joint Workshop on Health Intelligence*, 2018.

[35] J. Lawrence and C. Reed, Argument mining: A survey, *Computational Linguistics* **45**(4) (2020), 765–818. doi:10.1162/coli_a_00364.

[36] M. Lippi and P. Torroni, Argumentation mining: State of the art and emerging trends, *ACM Transactions on Internet Technology (TOIT)* **16**(2) (2016), 1–25. doi:10.1145/2850417.

[37] N. Maudet, S. Parsons and I. Rahwan, Argumentation in multi-agent systems: Context and recent developments, in: *ArgMAS*, N. Maudet, S. Parsons and I. Rahwan, eds, Lecture Notes in Computer Science, Vol. 4766, Springer, 2006, pp. 1–16.

[38] P. Mcburney and S. Parsons, Games that agents play: A formal framework for dialogues between autonomous agents, *Journal of Logic, Language and Information* **11** (2002), 315–334.

[39] P. Mcburney and S. Parsons, Dialogue games in multi-agent systems, *Informal Logic* **22** (2002).

[40] P. McBurney and S. Parsons, Locutions for argumentation in agent interaction protocols, in: *AC*, R.M. van Eijk, M.-P. Huget and F. Dignum, eds, Lecture Notes in Computer Science, Vol. 3396, Springer, 2004, pp. 209–225.

[41] P. McBurney, S. Parsons et al., Argument schemes and dialogue protocols: Doug Walton's legacy in artificial intelligence, *Journal of Applied Logics* **8**(1) (2021), 263–286.

[42] S. Modgil and H. Prakken, The ASPIC+ framework for structured argumentation: A tutorial, *Argument & Computation* **5**(1) (2014), 31–62. doi:10.1080/19462166.2013.869766.

[43] S. Modgil and H. Prakken, The ASPIC+ framework for structured argumentation: A tutorial, *Argument & Computation* **5**(1) (2014), 31–62. doi:10.1080/19462166.2013.869766.

[44] A.R. Panisson, A framework for reasoning and dialogue in multi-agent systems using argumentation schemes, PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2019.

[45] A.R. Panisson, M-arguments, in: *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2020.

[46] A.R. Panisson, A. Ali, P. McBurney and R.H. Bordini, Argumentation schemes for data access control, in: *Computational Models of Argument (COMMA)*, 2018, pp. 361–368.

[47] A.R. Panisson and R.H. Bordini, Knowledge representation for argumentation in agent-oriented programming languages, in: *2016 Brazilian Conference on Intelligent Systems, BRACIS*, 2016.

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

[49] A.R. Panisson and R.H. Bordini, Uttering only what is needed: Enthymemes in multi-agent systems, in: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1670–1672.

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

[51] A.R. Panisson, D.C. Engelmann and R.H. Bordini, Engineering explainable agents: An argumentation-based approach, in: *International Workshop on Engineering Multi-Agent Systems (EMAS)*, 2021.

[52] A.R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira and R.H. Bordini, Arguing about task reallocation using ontological information in multi-agent systems, in: *12th International Workshop on Argumentation in Multiagent Systems*, 2015.

[53] A.R. Panisson, F. Meneguzzi, R. Vieira and R.H. Bordini, An approach for argumentation-based reasoning using defeasible logic in multi-agent programming languages, in: *11th International Workshop on Argumentation in Multiagent Systems*, 2014.
[54] A.R. Panisson, F. Meneguzzi, R. Vieira and R.H. Bordini, Towards practical argumentation-based dialogues in multi-agent systems, in: *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2015.
[55] A.R. Panisson, F. Meneguzzi, R. Vieira and R.H. Bordini, Towards practical argumentation in multi-agent systems, in: *2015 Brazilian Conference on Intelligent Systems, BRACIS 2015*, 2015.
[56] A.R. Panisson, S. Parsons, P. McBurney and R.H. Bordini, Choosing appropriate arguments from trustworthy sources, in: *Computational Models of Argument (COMMA)*, 2018, pp. 345–352.
[57] S. Parsons, K. Atkinson, K. Haigh, K. Levitt, P.M.J. Rowe, M.P. Singh and E. Sklar, Argument schemes for reasoning about trust, in: *Computational Models of Argument: Proceedings of COMMA 2012*, Vol. 245, 2012, p. 430.
[58] S. Parsons and P. McBurney, Argumentation-based dialogues for agent co-ordination, *Group Decision and Negotiation* **12**(5) (2003), 415–439. doi:10.1023/B:GRUP.0000003742.50038.d3.
[59] S. Parsons, M. Wooldridge and L. Amgoud, An analysis of formal inter-agent dialogues, in: *1st International Conference on Autonomous Agents and Multi-Agent Systems*, ACM Press, 2002, pp. 394–401.
[60] S. Parsons, M. Wooldridge and L. Amgoud, An analysis of formal inter-agent dialogues, in: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, AAMAS'02*, ACM, New York, NY, USA, 2002, pp. 394–401.
[61] G.D. Plotkin, A structural approach to operational semantics, 1981.
[62] H. Prakken, A formal model of adjudication dialogues, *Artificial Intelligence and Law* **16**(3) (2008), 305–328. doi:10.1007/s10506-008-9066-4.
[63] H. Prakken, An abstract framework for argumentation with structured arguments, *Argument and Computation* **1**(2) (2011), 93–124. doi:10.1080/19462160903564592.
[64] H. Prakken, A. Wyner, T. Bench-Capon and K. Atkinson, A formalization of argumentation schemes for legal case-based reasoning in ASPIC+, *Journal of Logic and Computation* **25**(5) (2015), 1141–1166. doi:10.1093/logcom/ext010.
[65] I. Rahwan and L. Amgoud, An argumentation based approach for practical reasoning, in: *AAMAS*, H. Nakashima, M.P. Wellman, G. Weiss and P. Stone, eds, ACM, 2006, pp. 347–354.
[66] I. Rahwan, C. Reed and F. Zablith, On building argumentation schemes using the argument interchange format, in: *Working Notes of the 7th Workshop on Computational Models of Natural Argument (CMNA 2007)*, Hyderabad, 2007.
[67] I. Rahwan and G.R. Simari, *Argumentation in Artificial Intelligence*, Vol. 47, Springer, 2009.
[68] A.S. Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, in: *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Agents Breaking Away: Agents Breaking Away, MAAMAW'96*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996, pp. 42–55.
[69] C. Reed, *Argument Technology for Debating with Humans*, Nature Publishing Group, 2021.
[70] C. Reed and G. Rowe, Araucaria: Software for argument analysis, diagramming and representation, *International Journal on Artificial Intelligence Tools* **13**(04) (2004), 961–979. doi:10.1142/S0218213004001922.
[71] C. Reed and D. Walton, Towards a formal and implemented model of argumentation schemes in agent communication, *Autonomous Agents and Multi-Agent Systems* **11**(2) (2005), 173–188. doi:10.1007/s10458-005-1729-x.
[72] C. Reed, S. Wells, J. Devereux and G. Rowe, AIF$^+$: Dialogue in the argument interchange format, *Frontiers in Artificial Intelligence and Applications* **172** (2008), 311.
[73] F. Sadri, F. Toni and P. Torroni, Logic agents, dialogues and negotiation: An abductive approach, in: *Proceedings AISB'01 Convention, AISB*, 2001.
[74] D. Schmidt, A.R. Panisson, A. Freitas, R.H. Bordini, F. Meneguzzi and R. Vieira, An ontology-based mobile application for task managing in collaborative groups, in: *Florida Artificial Intelligence Research Society Conference*, 2016.
[75] J.R. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.
[76] E.I. Sklar and M.Q. Azhar, Argumentation-based dialogue games for shared control in human-robot systems, *Journal of Human-Robot Interaction* **4**(3) (2015), 120–148. doi:10.5898/JHRI.4.3.Sklar.
[77] P. Tolchinsky, K. Atkinson, P. McBurney, S. Modgil and U. Cortés, Agents deliberating over action proposals using the ProCLAIM model, in: *International Central and Eastern European Conference on Multi-Agent Systems*, Springer, 2007, pp. 32–41.
[78] P. Tolchinsky, U. Cortés, J. Nieves, A. López-Navidad and F. Caballero, Using arguing agents to increase the human organ pool for transplantation, in: *Third Workshop on Agents Applied in Health Care*, 2005.
[79] P. Tolchinsky, S. Modgil, K. Atkinson, P. McBurney and U. Cortés, Deliberation dialogues for reasoning about safety critical actions, *Autonomous Agents and Multi-Agent Systems* **25**(2) (2012), 209–259. doi:10.1007/s10458-011-9174-5.
[80] F. Toni, A tutorial on assumption-based argumentation, *Argument & Computation* **5**(1) (2014), 89–117. doi:10.1080/19462166.2013.869878.

[81] A. Toniolo, F. Cerutti, N. Oren, T.J. Norman and K. Sycara, Making informed decisions with provenance and argumentation schemes, in: *Proceedings of the Eleventh International Workshop on Argumentation in Multi-Agent Systems, 2014*, 2014.
[82] A. Toniolo, T.J. Norman, A. Etuk, F. Cerutti, R.W. Ouyang, M. Srivastava, N. Oren, T. Dropps, J.A. Allen and P. Sullivan, Supporting reasoning with different types of evidence in intelligence analysis, in: *International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 781–789.
[83] S.E. Toulmin, *The Uses of Argument*, Cambridge University Press, 1958.
[84] B. Verheij, Dialectical argumentation with argumentation schemes: An approach to legal logic, *Artificial intelligence and Law* **11**(2–3) (2003), 167–195. doi:10.1023/B:ARTI.0000046008.49443.36.
[85] R. Vieira, A. Moreira, M. Wooldridge and R.H. Bordini, On the formal semantics of speech-act based communication in an agent-oriented programming language, *J. Artif. Int. Res.* **29**(1) (2007), 221–267.
[86] J. Visser, J. Lawrence and C. Reed, Reason-checking fake news, *Communications of the ACM* **63**(11) (2020), 38–40. doi:10.1145/3397189.
[87] D. Walton, *Argumentation Schemes for Presumptive Reasoning*, Routledge, 1996.
[88] D. Walton and E. Krabbe, *Commitment in Dialogue: Basic Concept of Interpersonal Reasoning*, State University of New York Press, Albany NY, 1995.
[89] D. Walton, C. Reed and F. Macagno, *Argumentation Schemes*, Cambridge University Press, 2008.
[90] D.N. Walton and L.M. Batten, Games, graphs and circular arguments, *Logique et Analyse* **27**(106) (1984), 133–164.
[91] S. Wells, Supporting argumentation schemes in argumentative dialogue games, *Studies in Logic, Grammar and Rhetoric* **36**(1) (2014), 171–191. doi:10.2478/slgr-2014-0009.
[92] M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2009.
[93] A. Wyner, A functional perspective on argumentation schemes, *Argument & Computation* **7**(2–3) (2016), 113–133. doi:10.3233/AAC-160010.