DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)

Departamento de Sistemas
Informáticos y Computación

Technical Report

Ref. No.:
Pages: 48
Title: A Computational Argumentation Framework for Agent Societies
Author(s): Stella Heras and Vicente Botti and Vicente Julián
Date: June 17, 2011
Keywords: Argumentation, Case-Based Reasoning, Multi-Agent Systems

V^o B^o
Vicente Botti Stella Heras

1 Introduction

In computing, large systems can be viewed or designed in terms of the services they offer and the entities that interact to provide or consume these services. This paradigm is known as computing as interaction [28]. Open Multi-Agent Systems (MAS), where agents can enter or leave the system, interact and dynamically form agent' coalitions or organisations to solve problems seems a suitable technology to implement this new paradigm. However, the high dynamism of the domains where open MAS commonly operate requires agents to have a way of reaching agreements that harmonise the conflicts that come out when they have to collaborate or coordinate their activities. In addition, agents in open MAS can form societies that link them via dependency relations. These relations can emerge from agents' interactions or be predefined by the system. Anyway, the dependencies between agents are part of their *social context*, which has an important influence in the way agents can reach agreements with other agents. To illustrate this idea, let us introduce an example scenario where a group of agents is arguing about the best allocation for a water-right transfer in a river basin. A water-right is a contract with the basin administration organism that specifies the rights that a user has over the water of the basin (e.g. the maximum volume that he can spend, the price that he must pay for the water or the district where the water is settled). The owner of a water-right is allowed to temporally sell this right if he is not making use of his right. In that case, the basin administrator and some potential buyers (e.g. farmers) can argue to reach an agreement about the final beneficiary of the transfer. In addition, different considerations can be taken into account to make this decision, such as fairness, ecological issues, preferences over users, etc. In this example, for instance, farmers could not be as willing to accept arguments from other farmers as they are from the basin administrator. Similarly, business companies are other common example scenario where the social context of agents has an important influence in the way they argue. Here, subordinates are not as willing to accept tasks from equals as they are from superiors. Also, workers do not behave in the same way if, for instance, they are negotiating their own salaries than if they act as representatives of their trade unions.

Argumentation provides a fruitful means of dealing with conflicts and knowledge inconsistencies [33]. However, little work, if any, has been done to study the effect of the social context of agents in the way that they argue. Commonly, the term *agent society* is used in the Argumentation and Artificial Intelligence (AI) literature as a synonym for an *agent organisation* [18] or a *group of agents* that play specific roles, follow some interaction patterns and collaborate to reach global objectives [30]. Our notion of agents' social context in an argumentation process (defined with more detail in Section 3), includes information related to the proponent and the opponent of an argument, the group that these agents belong to and the dependency relation that they have. In addition to the dependency relations between agents, they can also have *values*. These values can be *personal goods* that agents want to promote or demote (e.g. solidarity, peace, etc.) or also *social goods* inherited from the agents' dependency relations (e.g. in a negotiation, a superior could impose his values to his subordinates or, on the opposite, a trade unionist might have to adopt the values of the collective that he represents, which can be different from his own values). Thus, we endorse the view of value-based argumentation frameworks [9, 6], which stress the importance of the audience in determining whether an argument is persuasive or not. Value-based argumentation frameworks extend abstract argumentation frameworks by addressing issues about the rational justification of choices. In their definition, they introduce the concepts of *values* that arguments of an argumentation framework promote or demote and *audiences*, which are sets of individuals that share a particular preference order over the set of values. Then, we also consider values as an important element of the social context of an agent.

To our knowledge, few research has been done to adapt multi-agent argumentation frameworks to represent arguments of agents taking into account their social context. Starting from the idea that the social context of agents determines the way in which agents can argue and reach agreements, this context should have a decisive influence in the computational representation of arguments. In this report, we advance research in the area of computational frameworks for agent argumentation by proposing a new argumentation framework (AF) for the design of open MAS in which the participating software agents are able to manage and exchange arguments between themselves taking into account the agents' social context. In order to do this, we have analysed the necessary requirements for this type of framework and taken into account them in the design of our framework. Also, the knowledge resources that the agents can use to manage arguments in this framework are presented in this work. In addition, if heterogeneous agents can interact in the framework, they need a common language to represent arguments and argumentation processes. To cope with this, we have also designed an argumentation ontology to represent arguments and argumentation concepts in our framework.

The structure of the report is the following: Section 2 introduces our concept of agent society and analyses the requirements that an argumentation framework for agent societies should met; Section 3 shows the knowledge resources proposed in our framework; Section 4 provides a formal specification for the argumentation framework; Section 5 shows an example scenario in the water-rights transfer domain; Section 6 reviews related work and discusses some assumptions made in this report and finally, Section 7 provides conclusions about the contributions of this research.

2 Requirements for an Argumentation Framework for Agent Societies

As the first step in our research, we have identified the necessary requirements for the framework proposed in this work, taking into account that we consider that agents belong to a society. Therefore, in this section we introduce the formal definition of the concepts that constitute our view of agent societies. Then, we analyse the issues that have been considered to choose a suitable argumentation framework for agent societies.

2.1 Society Model

In this work, we extend the approach of [4] and [14], who define an *agent society* in terms of a *group* of *agents* that play a set of *roles*, observe a set of *norms* and a set of *dependency relations* between roles and use a *communication language* to collaborate and reach global objectives. This definition is generic enough to fit most types of agent societies, such as social networks of agents [23] or open agent organisations [15, 18]. Broadly speaking, it can be adapted to any open MAS where there are norms that regulate the behaviour of agents, roles that agents play, a common language that allow agents to interact defining a set of permitted locutions and a formal semantics for each of these elements. Here, we differentiate the concept of agent society from the concept of a group of agents, in the sense that we consider that the society is the entity that establishes the dependency relations between the agents, which have values and can dynamically group together to pursue common objectives in a specific situation. Similarly, our approach of an agent society differs from the notion of agent *coalitions* used in [2] and [11], which are temporary associations between agents in order to carry out joint tasks, without any consideration about the social links and values that characterise those agents.

However, we consider that the values that individual agents or groups want to promote or demote and preference orders over them have also a crucial importance in the definition of an argumentation framework for agent societies. These values could explain the reasons that an agent has to give preference to certain beliefs, objectives, actions, etc. Also, dependency relations between roles or the membership to different heterogeneous groups could imply that an agent must change or violate its own value preference order. For instance, an agent of higher hierarchy could impose its values to a subordinate or an agent could have to adopt a certain preference order over values to be accepted in a group. Therefore, we endorse the view of [6], [34] and [41], who stress the importance of the audience, which has a preference order over values, in determining whether an argument (e.g. for accepting or rejecting someone else's beliefs, objectives or action proposals) is persuasive or not. Thus, we have included in the above definition of agent society the notion of values and preference orders among them. Next, we provide a formal definition for the model of society that we have adopted:

Definition 2.1 (Agent Society)**.** *An Agent society in a certain time t is defined as a tuple $S_t = <$ Ag, Rl, D, G, N, V, Roles, Dependency, Group, val, $Valpref_Q >$ where:*

- $Ag = \{ag_1, ag_2, ..., ag_I\}$ *is a finite set of I agents members of S_t in a certain time t.*

- *$Rl = \{rl_1, rl_2, ..., rl_J\}$ is a finite set of J roles that have been defined in S_t.*

- *$D = \{d_1, d_2, ..., d_K\}$ is a finite set of K possible dependency relations among roles defined in S_t.*

- *$G = \{g_1, g_2, ..., g_L\}$ is a finite set of groups that the agents of S_t form, where each $g_i, 1 \leq i \leq L, g_i \in G$ consist of a set of agents $a_i \in Ag$ of S_t.*

- *N is a finite set of norms that affect the roles that the agents play in S_t.*

- *$V = \{v_1, v_2, ..., v_P\}$ is a set of P values predefined in S_t.*

- *$Roles : Ag \rightarrow 2^{Rl}$ is a function that assigns an agent its roles in S_t.*

- *$Dependency_{S_t} : \forall d \in D, <_d^{S_t} \subseteq Rl \times Rl$ defines a reflexive, symmetric and transitive partial order relation over roles.*

- *$Group : Ag \rightarrow 2^G$ is a function that assigns an agent its groups in S_t.*

- *$val : Ag \rightarrow V$ is a function that assigns an agent the set of values that it has.*

- *$Valpref_Q : \forall q \in Ag \cup G, <_q^{S_t} \subseteq V \times V$ defines a reflexive, symmetric and transitive partial order relation over the values of an agent or a group.*

That is, $\forall r_1, r_2, r_3 \in R, r_1 <_d^{S_t} r_2 <_d^{S_t} r_3$ implies that r_3 has the highest rank with respect to the dependency relation d in S_t. Also, $r_1 <_d^{S_t} r_2$ and $r_2 <_d^{S_t} r_1$ implies that r_1 and r_2 have the same rank with respect to d. Finally, $\forall v_1, v_2, v_3 \in V, Valpref_a = \{v_1 <_a^{S_t} v_2 <_a^{S_t} v_3\}$ implies that agent a prefers value v_3 to v_2 and value v_2 to value v_1 in S_t. Similarly, $Valpref_g = \{v_1 <_g^{S_t} v_2 <_g^{S_t} v_3\}$ implies that group g prefers value v_3 to v_2 and value v_2 to value v_1 in S_t.

Once the concepts that we use to define agent societies are specified, the next section analyses the computational requirements for argument representation in these societies.

2.2 Computational Requirements for Arguments in Agent Societies

The previous section has introduced our view of agent societies. Now, this section studies the requirements that an argumentation framework for agent societies should met. An argumentation process is conceived as a reasoning model with several steps [3]:

1. Building arguments (supporting or attacking conclusions) from knowledge bases.

2. Defining the strengths of those arguments.

3. Determining the different conflicts between arguments.

4. Evaluating the acceptability of arguments in view of the other arguments that are posed in the dialogue.

5. Defining the justified conclusions of the argumentation process.

The first step to design MAS whose agents are able to perform argumentation processes is to decide how agents represent arguments. According to the *interaction problem* defined in [12], *"...representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem..."*. Therefore the way in which agents computationally represent arguments should ease the automatic performance of argumentation processes.

Some research effort on the computational representation of arguments is performed in the area of developing models for argument authoring and diagramming. Important contributions are [38, 40] and the Online Visualisation of Argument (OVA[a]) project. However, these systems assume human users interacting with the software tool and are not conceived for performing agents' automatic reasoning

[a]`www.arg.dundee.ac.uk`

processes. Other research works where the computational modelling of arguments has been studied are those on case-based argumentation. From the first uses of argumentation in AI, arguments and cases are intertwined [42]. Case-based argumentation particularly reported successful applications in American common law [7], whose judicial standard orders that similar cases must be resolved with similar verdicts. In [8] a model of legal reasoning with cases is proposed. But, again, this model assumed human-computer interaction and cases were not thought to be only acceded by software agents. Case-Based Reasoning (CBR) systems [1] allow agents to learn from their experiences. In MAS, the research in case-based argumentation is quite recent with just a few proposals to date [21]. These proposals are highly domain-specific (e.g. persuasion in negotiation [44], sensor networks [43] and classification [31]) or centralise the argumentation functionality either in a *mediator* agent, which manages the dialogue between the agents of the system [46], or in a specific module of the system itself [24].

As pointed out before, we focus on argumentation processes performed among a set of agents that belong to an agent society and must reach an agreement to solve a problem taking into account their social dependencies. Each agent builds its individual position in view of the problem (a solution for it). At this level of abstraction, we assume that this could be a generic problem of any type (e.g. a resource allocation problem, an agreed classification, a joint prediction, etc.) that could be characterised with a set of features. Thus, we assume that each agent has its individual knowledge resources to generate a potential solution. Also, agents have their own argumentation system to create arguments to support their positions and defeat the ones of other agents.

Taking into account the above issues, we have identified a set of requirements that a suitable framework to represent arguments in agent societies should met:

- be computationally tractable and designed to ease the performance of automatic reasoning processes over it.
- be rich enough to represent general and context dependent knowledge about the domain and social information about the agents' dependency relations or the agents' group.
- be generic enough to represent different types of arguments.
- comply with the technological standards of data and argument interchange on the web.

These requirements suggest that an argumentation framework for agent societies should be easily interpreted by machines and have highly expressive formal semantics to define complex concepts and relations over them. Thus, we propose a Knowledge-Intensive (KI) case-based argumentation framework [13], which allows automatic reasoning with semantic knowledge in addition to the syntactic properties of cases. Reasoning with cases is especially suitable when there is a weak (or even unknown) domain theory, but acquiring examples encountered in practice is easy. Most argumentation frameworks and their resulting argumentation systems produce arguments by applying a set of inference rules. This is the case of the recently proposed ASPIC framework [35], which is an abstract model of rule-based argumentation with structured arguments.

Rule-based systems require to elicit a explicit model of the domain. In open MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance, even if they are *domain-specific inference rules* intended to represent domain knowledge. However, tracking the arguments that agents put forward in argumentation processes could be relatively simple. Therefore, these arguments can be stored as cases codified in a specific case representation language that different agents are able to understand. This is easier than creating an explicit domain model, as it is possible to develop case-bases avoiding the knowledge-acquisition bottleneck. With these case-bases, agents are able to perform *lazy learning* processes over argumentation information, as will be illustrated in the example of Section 5. Another important problem with rule-based systems arises when the knowledge-base must be updated (e.g. adding new knowledge that can invalidate the validity of a rule). Updates imply to check the knowledge-base for conflicting or redundant rules. Case-based systems are easier to maintain than rule-based systems since, in the worst case, the addition of new cases can give rise to updates in some previous cases, but does not affect the correct operation of the system, although it can have an impact in its performance. Hence, a case-based representation of the domain knowledge of the system is more suitable for being applied in dynamic open MAS.

Next section makes a proposal for the type of knowledge resources that agents in agent societies could have to generate, select and evaluate positions and arguments taking into account the identified requirements.

3 Knowledge Resources

In open multi-agent argumentation systems the arguments that an agent generates to support its position can conflict with arguments of other agents and these conflicts are potentially solved by means of argumentation dialogues between them (since a dialogue may still terminate in disagreement). In our framework we propose three types of knowledge resources that the agents can use to generate, select and evaluate arguments in view of other arguments:

- Domain-cases database, with domain-cases that represent previous problems and their solutions. The structure of these cases is domain-dependent and thus is not detailed in this report.
- Argument-cases database, with argument-cases that represent previous argumentation experiences and their final outcome.
- Argumentation schemes [48], with a set of argumentation schemes, which represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. An argumentation scheme consists of a set of premises and a conclusion that is presumed to follow from them. Also, each argumentation scheme has associated a set of *critical questions* that represent potential attacks to the conclusion supported by the scheme. The concrete argumentation schemes to be used depend on the application domain.

In addition, arguments in our framework can be attacked by putting forward the following attack elements:

- Critical questions: when the conclusion of the argument was drawn by using an argumentation scheme, this conclusion can be attacked by posing a critical question attached to this scheme.
- Distinguishing premises: which are premises that can invalidate the application of a knowledge resource to generate a valid conclusion for an argument.
- Counter-examples: which are cases that are similar to a case (their descriptions match) but have different conclusions.

Next, these attack elements are described formally. First, let us introduce some functions that are used in the definitions. Let F be a set of features, C a set of cases and V a set of values.

Definition 3.1 (Function Value)**.** *The function value is defined as* $value_k(x) : C \times F \rightarrow V$ *and returns for a case* $k \in C$ *the value of the feature* $x \in F_k$ *(from a set of features* F_k *of the case* k*).*

Definition 3.2 (Match)**.** *A match between two sets of features* $i, j \in F$ *is defined as:* $match(i,j) : F \times F \rightarrow true$ *iff* $F_i \cap F_j \neq \emptyset$ *and* $\forall f \in F_i \cap F_j,\ val_i(f) = val_j(f)$*.*

Hence, two cases match if the features that appear in both cases have the same value for them. Note that this does not mean that both cases have the same features, since any of them can have extra features that do not appear in the other case.

Definition 3.3 (Subsumption)**.** *A case* $c_l \in C$ *subsumes other case* $c_m \in C$*:* $subsumes(c_l, c_m) : C \times C \rightarrow true$ *iff* $match(c_l, c_m)$ *and* $\forall f_m \in c_m, \exists f_l \in c_l / val_{c_l}(f_l) = val_{c_m}(f_m)$*.*

Therefore, we also describe problems as cases without solution and assume that a match between the problem to solve and a stored case means that the latter has some features of the problem and with the same values[b]. A total match between a problem and a case or between two cases means that both cases have the same features and with the same values. Now, based on the above definitions we have that:

Definition 3.4 (Counter-Example). *A counter-example for a case is a previous domain-case or an argument-case that was deemed acceptable, where the problem description of the counter-example matches the current problem to solve and also subsumes the problem description of the case, but proposing a different solution.*

Definition 3.5 (Distinguishing Premise). *A distinguishing premise x with respect to a problem P between two cases $c_1, c_2 \in C$ is defined as: $\exists x \in c_1 \wedge \nexists x \in P \ / \ \exists x \in c_2 \wedge value_{c_1}(x) \neq value_{c_2}(x)$ or else, $\exists x \in c_1 \wedge \exists x \in P \ / \ value_{c_1}(x) = value_P(x) \wedge \nexists x \in c_2$, where $P \subseteq F$, $x \in F$ and $c_1, c_2 \in C$.*

That is a premise that does not appear in the problem description and has different values for two cases or a premise that appears in the problem description that does not appear in one of the cases. Note that distinguishing premises are sometimes implicit in counter-examples, when the counter-example has features that do not appear in the original description of the problem to solve or in the description of the case that the counter-example rebuts.

In this section we describe the knowledge resources introduced above by using an ontological approach that observes the requirements put forward in the last section (see appendix for an explanation about the notation used). Thus, this ontology provides a common language to represent the resources and it is computationally tractable, rich enough to represent different types of domain-specific and general knowledge, generic enough to represent different types of arguments and compliant with the technological standards of data and argument interchange in the Web.

3.1 ArgCBROnto Ontology: General Concepts

We have designed an ontology called *ArgCBROnto* to define the representation language of the above knowledge resources. Ontologies provide a common vocabulary to understand the structure of information among different software agents. In addition, ontologies allow to make assumptions about the domain explicit, which facilitates to change these assumptions as new knowledge about the domain is acquired. As explained before, the high dynamism of the domains where open MAS commonly operate gives rise to many changes in the domain knowledge that agents have available and they must be able to handle the consequences of these changes. Thus, the vocabulary of domain-cases, argument-cases and argumentation schemes is defined by using the ArgCBROnto ontology. In addition, the ArgCBROnto ontology follows the approach of the case-based KI systems proposed in [13]. KI-CBR enables automatic reasoning with semantic knowledge in addition to the syntactic properties of cases. This allows one to make semantic inferences with the elements of cases and use more complex measures to compute the similarity between them. Next, we provide a general view of the ArgCBROnto ontology for the argumentation framework proposed in this report, with focus on the concepts that define the knowledge resources presented in this section.

In the top level of abstraction, the terminological part of the ontology distinguishes between several disjoint concepts. Among them we have the concepts of *Case*, which is the basic structure to store the argumentation knowledge of agents; *CaseComponent*, which represents the usual parts that cases have in CBR systems and *ArgumentationScheme*, which represents the argumentation schemes that the framework has.

$Case \sqsubseteq Thing \qquad Case \sqsubseteq \neg CaseComponent$

$CaseComponent \sqsubseteq Thing \qquad CaseComponent \sqsubseteq \neg ArgumentationScheme$

[b]Different types of matches could define other types of similarity between cases. For instance, a different match function could establish the threshold under which two features can be considered as similar or when a feature subsumes other feature in a hierarchy (and hence the more specific feature could be considered as a matching feature).

$ArgumentationScheme \sqsubseteq Thing \quad ArgumentationScheme \sqsubseteq \neg Case$

As pointed out before, there are two disjoint types of cases, *domain-cases* and *argument-cases* (see Figure 1).

$ArgumentCase \sqsubseteq Case \quad DomainCase \sqsubseteq Case$

$ArgumentCase \sqsubseteq \neg DomainCase$

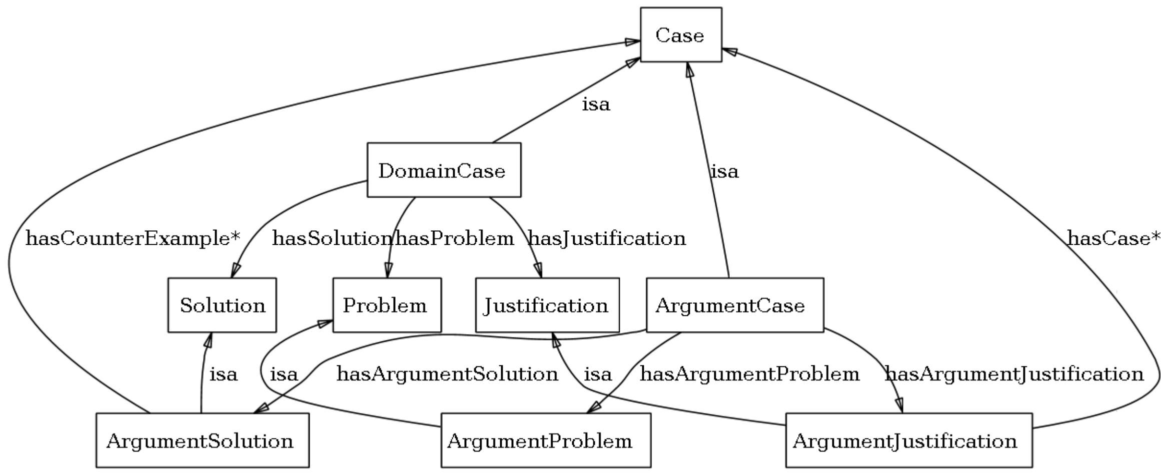


Figure 1: ArgCBROnto Case

Cases have the three possible types of components that usual cases of CBR systems have (see Figure 2): the description of the state of the world when the case was stored (*Problem*); the solution of the case (*Conclusion*); and the explanation of the process that gave rise to this conclusion (*Justification*). These concepts are disjoint.

$Problem \sqsubseteq CaseComponent \quad Solution \sqsubseteq CaseComponent$

$Justification \sqsubseteq CaseComponent$

$Problem \sqsubseteq \neg Solution\ Conclusion \sqsubseteq \neg Justification$

$Problem \sqsubseteq \neg Justification$

Domain-cases have the usual problem, conclusion and justification parts, as shown in Figure 1.

$DomainCase \sqsubseteq \forall hasProblem.Problem$

$DomainCase \sqsubseteq \forall hasSolution.Solution$

$DomainCase \sqsubseteq \forall hasJustification.Justification$

However, argument-cases have a more specialised description for these components (*ArgumentProblem*, *ArgumentSolution* and *ArgumentJustification*), which includes an extended set of properties (see Figure 1).

$ArgumentProblem \sqsubseteq Problem\ ArgumentSolution \sqsubseteq Solution$

$ArgumentJustification \sqsubseteq Justification$

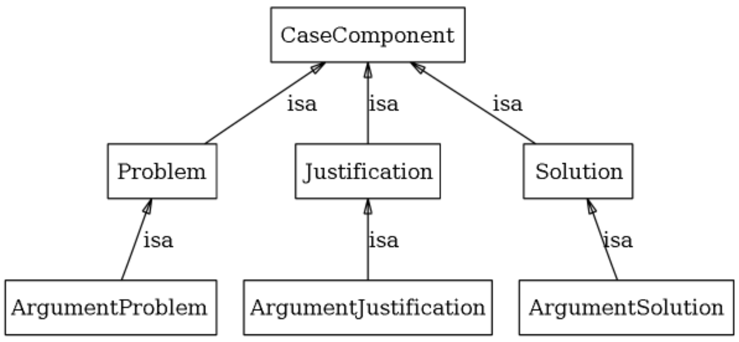


Figure 2: ArgCBROnto CaseComponent

$ArgumentCase \sqsubseteq \forall hasArgumentProblem.ArgumentProblem$

$ArgumentCase \sqsubseteq \forall hasArgumentSolution.ArgumentSolution$

$ArgumentCase \sqsubseteq \forall hasArgumentJustification.ArgumentJustification$

Also, cases have as properties a unique identifier *ID* and a *creation date*, with its corresponding range and domain.

$\top \sqsubseteq \forall hasID.Integer$

$\top \sqsubseteq \forall\ hasID^{-}.(\text{Case} \sqcup \text{SocialEntity} \sqcup \text{Value} \sqcup \text{Norm} \sqcup \text{Argument} \sqcup \text{ArgumentationScheme} \sqcup \text{Premise})$[c]

$\top \sqsubseteq \forall\ \text{hasCreationDate.Date}\ \top \sqsubseteq \forall hasCreationDate^{-}.(\text{Case} \sqcup \text{ArgumentationScheme})$[c]

As pointed out before section, argumentation schemes represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. Each argumentation scheme consists of a set of *premises*, a *conclusion* drawn from these premises and a set of *critical questions* that represent potential attacks to the conclusion supported by the scheme. These critical questions can be classified as *presumptions* that the proponent of the argumentation scheme has made or *exceptions* to the general inference rule that the scheme represents [36]. In the former case, the proponent has the burden of proof if the critical question is asked, whereas in the later the burden of proof falls on the opponent that has questioned the conclusion of the scheme. Figure 3 shows the representation of an argumentation scheme in the ArgCBROnto ontology.

$ArgumentationScheme \sqsubseteq Thing$

$ArgumentationScheme \sqsubseteq \forall hasPremise.Premise$

$ArgumentationScheme \sqsubseteq \forall hasConclusion.Conclusion$

$ArgumentationScheme \sqsubseteq \forall hasPresumption.Premise$

$ArgumentationScheme \sqsubseteq \forall hasException.Premise$

In addition, for each argumentation scheme the ArgCBROnto ontology stores information about its unique *ID* (which ontological definition was provided before in this section), its *title*, its *creation date* and its *author*.

[c]Note that this property has as domain several concepts of the ArgCBROnto ontology, some of which will be introduced later in this article.

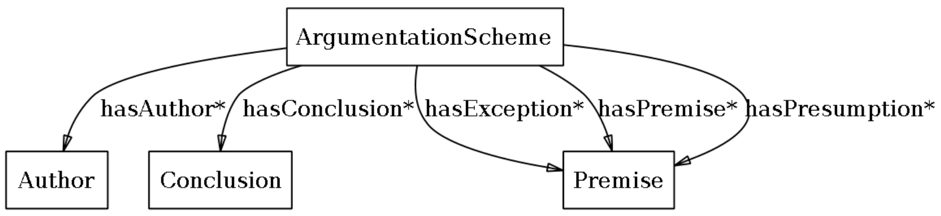


Figure 3: ArgCBROnto ArgumentationScheme

$\top \sqsubseteq \forall argTitle.String$

$\top \sqsubseteq \forall argTitle^{-}.ArgumentationScheme$

$\top \sqsubseteq \forall creationDate.Date$

$\top \sqsubseteq \forall creationDate^{-}.ArgumentationScheme$

$ArgumentationScheme \sqsubseteq \forall hasAuthor.Author$

The argument-cases are the main structure that we use to implement our framework and computationally represent arguments in agent societies. Also, their structure is generic and domain-independent. Thus, next section presents the ontological description for argument-cases in detail.

3.2 Argument-case Description

Argument-cases have two main objectives:

1. They can be used by agents as knowledge resources to generate new arguments and to select the best position to put forward in view of past argumentation experiences.
2. They can be used to store new argumentation knowledge that agents gain in each dialogue, improving the agents' argumentation skills.

Table 1 shows the structure of a generic argument-case. As pointed out before, the argument-cases have three main parts: the description of the *problem* that the case represents, the *solution* applied to this problem and the *justification* why this particular solution was applied. An argument-case stores the information about a previous argument that an agent posed in certain step of a dialogue with other agents.

Problem:

The problem description has a *domain context* that consists of the *premises* of the argument and represents the context of the domain where the argument was put forward. Each premise has a unique identifier *ID* (the ontological definition is provided at the end of Section 3.1), a *name* and a *content*, which can be of several types depending on the application domain.

$Context \sqsubseteq Thing$

$DomainContext \sqsubseteq Context$

$Problem \sqsubseteq \forall hasDomainContext.DomainContext$

$Premise \sqsubseteq Thing$

PROBLEM	Domain Context	[Premises]*	
	Social Context	Proponent	ID
			Role
			Norms
			ValPref
		Opponent	ID
			Role
			Norms
			ValPref
		Group	ID
			Role
			Norms
			ValPref
		Dependency Relation	
SOLUTION	Argument Type		
	Conclusion		
	Value		
	Acceptability Status		
	Received Attacks	[Critical Questions]*	
		[Distinguishing Premises]*	
		[Counter Examples]*	
JUSTIFICATION	[Cases]*		
	[Argumentation Schemes]*		
	Associated Dialogue Graph		

Table 1: Structure of an Argument-Case.

$\top \sqsubseteq \forall hasName.String \qquad \top \sqsubseteq \forall hasName^{-}.Premise$

$\top \sqsubseteq \forall hasContent.Type \qquad \top \sqsubseteq \forall hasContent^{-}.Premise$

In addition, if we want to store an argument and use it to generate a persuasive argument in the future, the features that characterise the audience of the previous argument (the *social context*) must also be kept. Thus, we have two disjoint types of contexts in our ontology, the usual domain context and the social context (shown in Figure 4).

$SocialContext \sqsubseteq Context$

$DomainContext \sqsubseteq \neg SocialContext$

$ArgumentProblem \sqsubseteq \forall hasSocialContext.SocialContext$

For the definition of the social context of arguments, we follow our model of society presented in Section 2.1. Therefore, we store in the argument-case the social information about each *social entity* related with the argument. This social entity can be an *agent* (the proponent of the argument and the opponent to which the argument is addressed) or else the *group* to which agents belong. Figure 5 shows this part of the ArgCBROnto ontology.

$SocialEntity \sqsubseteq Thing$

$Agent \sqsubseteq SocialEntity \qquad Group \sqsubseteq SocialEntity$

$Agent \sqsubseteq \neg Group$

For the sake of simplicity, in this report we assume that in each step of the dialogue, one proponent agent generates an argument and sends it to one opponent agent that belongs to its same group. However,

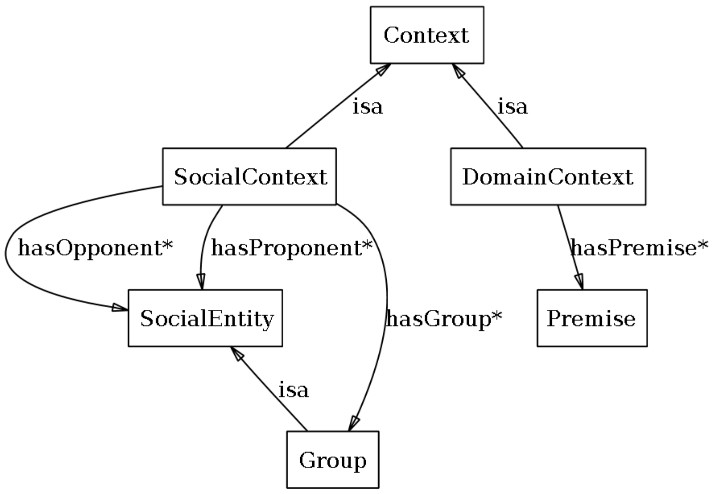


Figure 4: ArgCBROnto Context

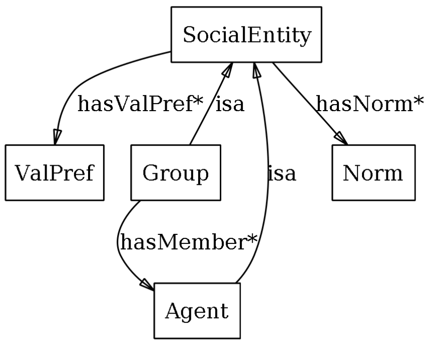


Figure 5: ArgCBROnto SocialEntity

either the proponent or the opponent's features could represent information about agents that act as representatives of a group and any agent can belong to different groups at the same time. Thus, the social context of argument-cases include information about the *proponent* and the *opponent* of the argument (which can be an agent or a group) and information about their *group*. Also, groups are formed by at least two agents.

$SocialContext \sqsubseteq \forall hasProponent.(Agent \sqcup Group)$

$SocialContext \sqsubseteq \forall hasOpponent.(Agent \sqcup Group)$

$SocialContext \sqsubseteq \forall hasGroup.Group$

$Group \sqsubseteq \forall hasMember.Agent \qquad Group \sqsubseteq \geq 2 hasMember$

Concretely, each social entity of the argument-case has a unique *ID* that identifies it in the system (the ontological definition is provided at the end of Section 3.1) and the *role* that the agent or the group was playing when it sent or received the argument (e.g. trade unionist, business manager, etc, do not confuse with the role of proponent and opponent from the argumentation perspective).

$\top \sqsubseteq \forall \mathit{hasRole}.\mathit{String} \qquad \top \sqsubseteq \forall \mathit{hasRole}^{-}.\mathit{SocialEntity}$

In addition, for each social entity a reference to the set of *norms* that governed the behaviour of the agents at this step of the dialogue is also stored, since the normative context of agents could force or forbid them to accept certain facts and the arguments that support them (e.g. a norm could invalidate a dependency relation or a value preference order). Also, each norm has a unique *ID* that identifies it (the ontological definition is provided at the end of Section 3.1) and a *description* with the semantics of the norm.

$Norm \sqsubseteq \mathit{Thing}$

$\mathit{SocialEntity} \sqsubseteq \forall hasNorm.Norm$

$\top \sqsubseteq \forall \mathit{hasDescription}.\mathit{String}$

$\top \sqsubseteq \forall \mathit{hasDescription}^{-}.(Conclusion \sqcup Value \sqcup Norm \sqcup Justification)$[c]

Moreover, if known, we also store the preferences of each agent or group over the pre-defined set of general *values* in the system (e.g. security, solidarity, economy, etc.). As pointed out before, these preferences (*ValPref*) affect the persuasive power of the proponent's argument over the opponent's behaviour. In the case of the group, we use this feature to store its *social values*[d].

$Value \sqsubseteq \mathit{Thing}$

$ValPref \sqsubseteq \mathit{Thing}$

$ValPref \sqsubseteq \forall hasPreferred.Value$

$\mathit{SocialEntity} \sqsubseteq \forall hasValPref.ValPref$

Finally, the *dependency relation* between the proponent's and the opponent's roles is also stored in the social context of the argument-cases. To date, we define the possible dependency relations between roles as in [14]:

- *Power*: when an agent has to accept a request from another agent because of some pre-defined domination relationship between them (e.g. in a society S_t that manages the water of a river basin, $Farmer <^{S_t}_{Power} \mathit{BasinAdministrator}$, since farmers must comply with the laws announced by the basin administrator [e]).
- *Authorisation*: when an agent has committed itself to another agent for a certain service and a request from the latter leads to an obligation when the conditions are met (e.g. in the society S_t, $Farmer_i <^{S_t}_{Authorisation} Farmer_j$, if $Farmer_j$ has contracted a service that offers $Farmer_i$).
- *Charity*: when an agent is willing to answer a request from other agent without being obliged to do so (e.g. in the society S_t, by default $Farmer_i <^{S_t}_{Charity} Farmer_j$).

Therefore, in our ArgCBROnto ontology we have these three types of dependency relations:

$\top \sqsubseteq \forall \mathit{hasDependencyRelation}.(\mathit{Power} \sqcup \mathit{Authorisation} \sqcup Charity)$

$\top \sqsubseteq \forall \mathit{hasDependencyRelation}^{-}.\mathit{SocialContext}$

[d]We use the term social values to refer those values that are agreed by (or commanded to) the members of a society as the common values that this society should promote (e.g. justice and solidarity in an ideal society) or demote.

[e]This example will be explained in detail in Section 5.

Solution:

In the solution part, the *conclusion* of the case (for both domain-cases and argument-cases) and the *value* promoted in this specific situation are stored (see Figure 6).

$Conclusion \sqsubseteq Thing$

$Solution \sqsubseteq \forall hasConclusion.Conclusion$

$Solution \sqsubseteq \forall promotesValue.Value$

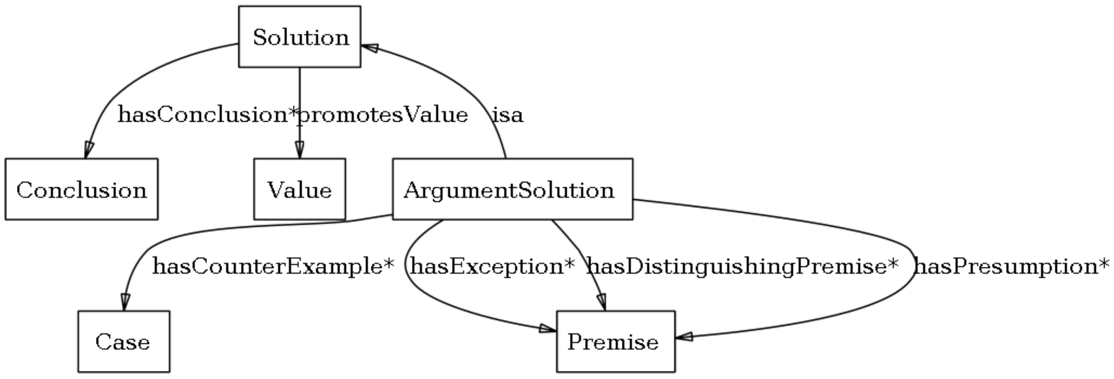


Figure 6: ArgCBROnto Solution

Also, for argument-cases we have a more specialised description for the solution part (*ArgumentSolution*), including the *argument type* that defines the method by which the conclusion of the argument was drawn is stored. By default, we do not assume that agents have a pre-defined set of rules to infer deductive arguments from premises, which is difficult to maintain in open MAS. In our framework, agents have the following ways of generating new arguments:

- *Presumptive arguments*: by using the premises that describe the problem to solve and an argumentation scheme whose premises match them.
- *Inductive arguments*: by using similar argument-cases and/or domain-cases stored in the case-bases of the system.
- *Mixed arguments*: by using premises, cases and argumentation schemes.

$ArgumentSolution \sqsubseteq Solution$

$\top \sqsubseteq \forall hasArgumentType.(Inductive \sqcup Presumptive \sqcup Mixed)$

$\top \sqsubseteq \forall hasArgumentType^{-}.ArgumentSolution$

Moreover, the solution part of the argument-cases stores the information about the *acceptability status* of the argument at the end of the dialogue. This feature shows if the argument was deemed *acceptable*, *unacceptable* or *undecided* in view of the other arguments that were put forward during the dialogue (see Section 4 for details).

$\top \sqsubseteq \forall hasAcceptabilityStatus.(Acceptable \sqcup Unacceptable \sqcup Undecided)$

$\top \sqsubseteq \forall hasAcceptabilityStatus^{-}.ArgumentSolution$

Regardless of the final acceptability status of the argument, the argument-case also stores in its solution part the information about the possible *attacks* that the argument received. These attacks could represent the justification for an argument to be deemed unacceptable or else reinforce the persuasive power of an argument that, despite being attacked, was finally accepted. Argument-cases can store different types of attacks, depending on the type of argument that they represent:

- For presumptive arguments: critical questions (*presumptions* or *exceptions*) associated with the scheme [48].
- For inductive arguments, as proposed in [8], either:
  - Premises which value in the context where the argument was posed was different (or non-existent) than the value that it took in the cases used to generate the argument (*distinguishing premises*) or
  - Cases which premises also match the premises of the context where the argument was posed, but which conclusion is different than the conclusion of the case(s) used to generate the argument (*counter-examples*).
- For mixed arguments: any of the above attacks.

Thus, the ArgCBROnto ontology represents the different types of attacks that arguments can receive as follows:

$ArgumentSolution \sqsubseteq \forall hasPresumption.Premise$

$ArgumentSolution \sqsubseteq \forall hasException.Premise$

$ArgumentSolution \sqsubseteq \forall hasDistinguishingPremise.Premise$

$ArgumentSolution \sqsubseteq \forall hasCounterExample.Case$

Justification:

In the ArgCBROnto ontology, the justification part of a case stores a *description* that can explain why this particular solution was applied to solve the case and what was the final results achieved. Figure 7 shows the concepts of the ArgCBROnto ontology that are related with this part of argument-cases.

$\top \sqsubseteq \forall hasDescription.String$

$\top \sqsubseteq \forall hasDescription^{-}.Justification$

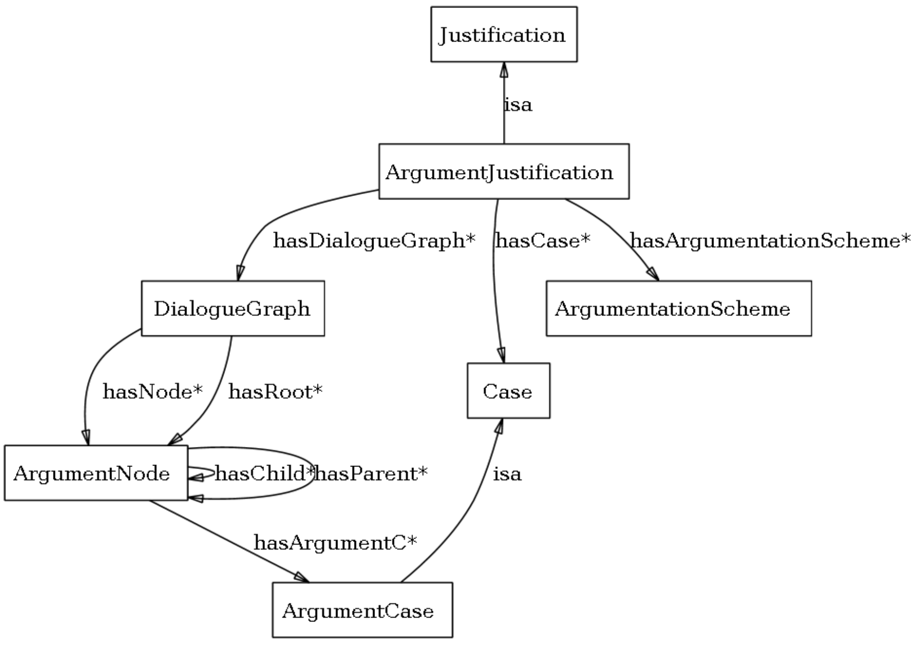


Figure 7: ArgCBROnto Justification

In the special case of argument-cases, the justification specialises in an *ArgumentJustification*, which stores the information about the knowledge resources that were used to generate the argument represented by the argument-case (e.g. the set *argumentation schemes* in presumptive arguments, the set of *cases* in inductive arguments and both in mixed arguments).

$ArgumentJustification \sqsubseteq Justification$

$ArgumentJustification \sqsubseteq \forall\ hasArgumentationScheme.ArgumentationScheme$

$ArgumentJustification \sqsubseteq \forall hasCase.Case$

In addition, the justification of each argument-case has associated a *dialogue-graph* that represents the dialogue where the argument was posed. The same dialogue graph can be associated with several argument-cases and an argument-case can be associated to several graphs. Each dialogue graph has a *root* and a set of nodes, which we call *argument nodes*. An argument node has an *argument-case*, a *parent* argument node and a *child* argument node. In this way, the ArgCBROnto ontology represents the sequence of arguments that were put forward in a dialogue, storing the complete conversation as a directed graph that links argument-cases. This graph can be used later to develop dialogue strategies. For instance, to improve efficiency in a negotiation an argumentation dialogue could be finished if it is being similar to a previous one that didn't reach an agreement. Alternatively, opponent moves in a dialogue (the arguments that it is going to present) could be inferred by looking a similar previous dialogue with the same opponent.

$DialogueGraph \sqsubseteq Thing$

$ArgumentNode \sqsubseteq Thing$

$ArgumentNode \sqsubseteq \forall hasArgumentC.ArgumentCase$

$ArgumentNode \sqsubseteq \forall hasParent.ArgumentNode$

$ArgumentNode \sqsubseteq \forall hasChild.ArgumentNode$

$DialogueGraph \sqsubseteq \forall hasRoot.ArgumentNode$

$DialogueGraph \sqsubseteq \forall hasNode.ArgumentNode$

$ArgumentJustification \sqsubseteq \forall hasDialogueGraph.DialogueGraph$

Following a CBR methodology, the proposed knowledge resources allow agents to automatically generate, select and evaluate arguments. However, the specification of this case-based reasoning process is out of the scope of this report. Here we have focused on defining how agents can represent arguments and argumentation related information to be able to perform an efficient and automatic management of this information. The argument-case structure presented is flexible enough to represent different types of arguments and their associated information. Also, the KI approach followed allows a semantic reasoning with the concepts that represent the cases. However, the value of some features on argument-cases and domain-cases could remain unspecified in some domains. For instance, in some open MAS, the preferences over values of other agents could not be previously known, although agents could try to infer the unknown features by using CBR adaptation techniques [27]. This and other open questions will be discussed in Section 6. The knowledge resources of our framework and the ArgCBROnto ontology that describes them have been presented in this section. Next section provides a formal definition for the framework.

4 Case-based Argumentation Framework for Agent Societies

Following our case-based computational representation of arguments, we have designed a formal Argumentation Framework for an Agent Society ($AFAS$) as an instantiation of Dung's argumentation framework [16]:

Definition 4.1 (Argumentation Framework for an Agent Society). *An argumentation framework for an agent society is a tuple* $AFAS = <A, R, S_t>$ *where:*

- A *is a set of arguments.*
- R *is an irreflexive binary attack relation on* A*.*
- S_t *is a society of agents as defined in Definition 2.1.*

The main advantages that our framework contributes over other existent AFs deal with the requirements suggested in Section 2.2. These advantages are: 1) the ability to represent social information in arguments; 2) the possibility of automatically managing arguments in agent societies; 3) the improvement of the agents' argumentation skills; and 4) the easy interoperability with other frameworks that follow the argument and data interchange web standards. According to Prakken [37], the elements that characterise an AF are: the notion of argument used in the framework, the logical language that represents argumentation concepts, the concept of conflict between arguments, the notion of defeat between arguments and the acceptability status of arguments. Next, the elements that characterise our case-based argumentation framework for agent societies are specified by using the knowledge resources defined in this section and the ArgCBROnto ontology.

4.1 The Notion of Argument: Case-Based Arguments

We have adopted the Argument Interchange Format (AIF) [52] view of arguments as a set of interlinked premiss-illative-conclusion sequences. The notion of argument is determined by our KI case-based framework to represent arguments. In our framework agents can generate arguments from the premises that represent the context of the domain where the argument is put forward in addition to the previous cases (domain-cases and argument-cases) and argumentation schemes used to generate the conclusion of the argument. Therefore, agents construct arguments by using their individual knowledge bases, which contain these types of knowledge resources.

Definition 4.2 (Knowledge Base)**.** *A Knowledge Base in a case-based argumentation framework for agent societies consists of a finite set of $\mathcal{K} \subseteq \mathcal{L}$ elements, where $\mathcal{L}$ is a logical language to represent these elements and $\mathcal{K} = \mathcal{K}_p \cup \mathcal{K}_{dc} \cup \mathcal{K}_{ac} \cup \mathcal{K}_{as}$, where each of these subsets are disjoint and:*

- $\mathcal{K}_p$ *is a finite set of premises.*
- $\mathcal{K}_{dc}$ *is a finite set of domain-cases.*
- $\mathcal{K}_{ac}$ *is a finite set of argument-cases.*
- $\mathcal{K}_{as}$ *is a finite set of argumentation schemes.*

Note that the fact that a proponent agent uses one or several knowledge resources to generate an argument does not imply that it has to show all this information to its opponent. The argument-cases of the agents' argumentation systems and the structure of the actual arguments that are interchanged between agents is not the same. Thus, arguments that agents interchange are defined as tuples of the form:

Definition 4.3 (Argument). *$Arg = \{\phi, v, < S >\}$, where ϕ is the conclusion of the argument, v is the value that the agent wants to promote with it and $< S >$ is a set of elements that support the argument (support set).*

In the ArgCBROnto ontology, in addition to the above elements (see Figure 8), arguments have a unique identifier *ID* (which ontological definition is provided at the end of Section 3.1):

$Argument \sqsubseteq Thing$

$SupportSet \sqsubseteq Thing$

$Argument \sqsubseteq \forall hasConclusion.Conclusion$

$Argument \sqsubseteq \forall promotesValue.Value$

$Argument \sqsubseteq \forall hasSupportSet.SupportSet$

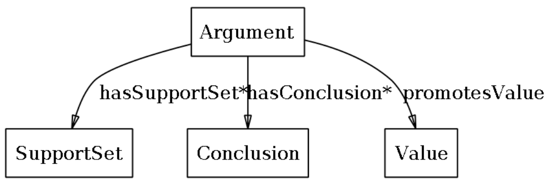


Figure 8: ArgCBROnto Argument

This support set can consist of different elements, depending on the argument purpose. On one hand, if the argument provides a potential solution for a problem, the support set is the set of features (*premises*) that represent the context of the domain where the argument has been put forward (those premises that match the problem to solve and other extra premises that do not appear in the description of this problem but that have been also considered to draw the conclusion of the argument) and optionally, any knowledge resource used by the proponent to generate the argument (*domain-cases*, *argument-cases* or *argumentation schemes*). On the other hand, if the argument attacks the argument of an opponent, the support set can also include any of the allowed attacks in our framework (*critical questions (presumptions and exceptions)*, *distinguishing premises* or *counter-examples*). Then, the support set consists of the following tuple of sets of support elements [f]:

Definition 4.4 (Support Set)**.** $S = <\{premises\}, \{domainCases\}, \{argumentCases\}, \{argumentationSchemes\}, \{criticalQuestions\}, \{distinguishingPremises\}, \{counterExamples\}>$

In the ArgCBROnto ontology, the elements of the support set are represented with the following properties (see Figure 9):

$SupportSet \sqsubseteq \forall hasPremise.Premise$

$SupportSet \sqsubseteq \forall hasDomainCase.DomainCase$

$SupportSet \sqsubseteq \forall hasArgumentCase.ArgumentCase$

$SupportSet \sqsubseteq \forall hasArgumentationScheme.ArgumentationScheme$

$SupportSet \sqsubseteq \forall hasPresumption.Premise$

$SupportSet \sqsubseteq \forall hasException.Premise$

$SupportSet \sqsubseteq \forall hasDistinguishingPremise.Premise$

$SupportSet \sqsubseteq \forall hasCounterExample.Case$

Therefore, arguments can be constructed by aggregating different support and attack elements, which are structures that support intermediate conclusions that lead to the conclusion of the argument itself. These elements can be viewed as the case-based version of the sub-arguments of a rule-based argumentation framework such as the ASPIC framework [35, Definition 3.6]). Inferences in our framework are

[f]This representation is only used for illustrative purposes and efficiency considerations about the implementation are obviated.

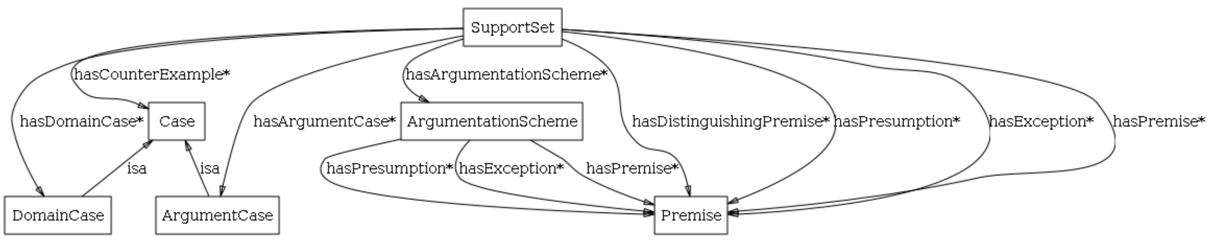


Figure 9: ArgCBROnto SupportSet

based on previous cases, on argumentation schemes and on premises (these that form part of the cases and argumentation schemes that match the problem to solve) instead of on strict or defeasible rules. An important difference between the definition of argument of our case-based approach and the ASPIC rule-based approach lies in the defeasible nature of all elements of the support set, except from certain premises, as explained below. By contrast, ASPIC arguments can be also constructed by chaining strict inference rules that give rise to indefeasible arguments.

Intuitively, arguments cannot be attacked on the support set premises that match the description of the problem to solve, but only on those extra premises that represent the context of the domain where the argument was put forward and that do not appear in the description of the problem. Therefore, the premises that describe the problem to solve can be considered as *axiom premises* as defined in the ASPIC framework [35, Definition 3.5]. Similarly, the extra premises can be considered as equivalent of the *ordinary premises* of the ASPIC framework and if the attacks on them result in defeats depend on the defeat relation specified below in Definition 4.8. Alternatively, the argument can be attacked on those premises that appear in the description of the problem to solve but have not been considered to draw the conclusion of the argument (do not appear in the support set of the argument). Again, compared with the ASPIC framework, these premises can be considered as *assumptions* and attacks on them always succeed.

In addition to the attacks on premises by means of distinguishing premises, our framework allows attacks on cases and argumentation schemes, which can be performed by means of counter-examples and critical questions respectively. If these attacks result or not in an argument defeat depend on the defeat relation of Definition 4.8.

4.2 The Logical Language

The logical language represents argumentation concepts and possible relations among them. In our framework, these concepts are represented in the form of KI cases and argumentation schemes. Therefore, the logical language of the AF is defined in terms of the vocabulary to represent these resources.

The vocabulary of cases and schemes is defined by using the ArgCBROnto ontology previously presented. We have selected the Ontology Web Language OWL-DL[g] as the logical language to represent the vocabulary of cases. This variant is based on Description Logics (DL) and guarantees computational completeness and decidability. Note that OWL-DL does not assume *closed world reasoning* with *negation as failure.* By contrast, OWL-DL uses *open world reasoning* with *negation as unsatisfiability.* Therefore, something is false only if it can be proved to contradict other information in the ontology. This implies that two *classes* (i.e. *concepts* in description logics terminology) are *contradictory* if and only if they are specifically declared as such with the owl property *complementOf* and similarly, two *instances* (i.e. *individuals* in description logics terminology) are contradictory if and only if they are specifically declared as such with the owl property *differentFrom.* This differs from the contrariness relation declared in the ASPIC framework [35], which also captures negation as failure. However, our restricted view of the contrariness relation follows the way of reasoning that case-based reasoning systems have.

Definition 4.5 (Logical Language)**.** *Let $\mathcal{L}$ be an OWL-DL language, C and D two owl classes, $c \in C$ and $d \in D$ two owl instances of the classes C and D respectively, and complementOf and differentFrom two contrariness functions from $\mathcal{L}$ to $2^{\mathcal{L}}$. Then, if C complementOf D (correspondingly c differentFrom d) and D complementOf C (d differentFrom c), C and D (c and d) are called contradictory. However, if C complementOf D (correspondingly c differentFrom d) and D $\neg$complementOf C (d $\neg$differentFrom c), C (c) is a contrary of D (d).*

To illustrate the difference between negation as failure and negation as unsatisfiability let us propose

[g] `www.w3.org/TR/owl-guide`

the following example (already introduced in Section 1): suppose that a basin administrator is arguing with a farmer to decide if, based on a case-base of previous experiences, the farmer should be the beneficiary of a water-right transfer. If the administrator cannot find a previous a case that in a similar situation granted the transfer to this (or a similar) user, a closed world reasoner would infer that the transfer is not appropriate (since the suitability of the transfer cannot be proved). By contrast, an OWL-DL reasoner that follows the open world assumption would infer that, in principle, there is no reason to approve or deny the transfer. Thus, if there are no other reasons that prevent the administrator to permit the transfer, it could finally approve it. This reflects the usual way of reasoning of the case-based reasoning methodology, which does not infer the negation of a conclusion by the mere fact that there is not a case in the case-base that supports this conclusion. OWL-DL allows automatic description logic reasoning over argument-cases and domain-cases. In addition, it facilitates the interoperability with other systems. In Sections 3 and 4.1, we have provided a partial view of the ontology for the case-based argumentation framework proposed[h].

4.3 The Concept of Conflict between arguments

The concept of conflict between arguments defines in which way arguments can attack each other. There are two typical attacks studied in argumentation: *rebut* and *undercut*. In an abstract definition, rebuttals occur when two arguments have contradictory conclusions. Similarly, an argument undercuts another argument if its conclusion is inconsistent with one of the elements of the support set of the latter argument or its associated conclusion. This section shows how our AF instantiates these two attacks. Taking into account the possible elements of the support set, rebut and undercut attacks can be formally defined as follows.

Let $Arg_1 = \{\phi_1, value_1, < S_1 >\}$ and $Arg_2 = \{\phi_2, value_2, < S_2 >\}$ be two different arguments, where $S_1 =< \{Premises\}_1, ..., \{CounterExamples\}_1 >$, $S_2 =< \{Premises\}_2, ..., \{CounterExamples\}_2 >$, $\sim$ stands for the logical negation, $\Rightarrow$ stands for the logical implication and $conc(x)$ is a function that returns the conclusion of the domain-case, argument-case or argumentation scheme x. Then:

[h]The complete specification of the ontology is available at `users.dsic.upv.es/~vinglada/docs`.

Definition 4.6 (Rebut)**.** *Arg_1 rebuts Arg_2 iff $\phi_1 = \sim\phi_2$ and $\{Premises\}_1 \supseteq \{Premises\}_2$*

That is, if Arg_1 supports a different conclusion for a problem description that includes the problem description of Arg_2 then Arg_1 rebuts Arg_2.

Definition 4.7 (Undercut)**.** *Arg_1 undercuts Arg_2 if*

$1)\phi_1 = \sim conc(as_k)/$

$\exists cq \in \{CriticalQuestions\}_1 \wedge \exists as_k \in \{ArgumentationSchemes\}_2 \wedge$

$cq \Rightarrow \sim conc(as_k)$, *or*

$2)\phi_1 = dp/$

$(\exists dp \in \{DistinguishingPremises\}_1 \wedge \exists pre_k \in \{Premises\}_2 \wedge dp = \sim pre_k) \vee$

$(dp \notin \{Premises\}_2)$, *or*

$3)\phi_1 = ce/$

$(\exists ce \in \{CounterExamples\}_1 \wedge \exists dc_k \in \{DomainCases\}_2$

$\wedge\, conc(ce) = \sim conc(dc_k)) \vee$

$(\exists ce \in \{CounterExamples\}_1 \wedge$

$\exists ac_k \in \{ArgumentCases\}_2 \wedge conc(ce) = \sim conc(ac_k))$

That is, if the conclusion drawn from Arg_1 makes one of the elements of the support set of Arg_2 or its conclusion non-applicable in the current context of the argumentation dialogue. In case 1 Arg_1 undercuts Arg_2 by posing a critical question that attacks the conclusion of Arg_2, inferred by using an argumentation scheme. In fact, the set of critical questions of an argumentation scheme constitute the set of possible undercuts to the conclusion drawn from the scheme. In case 2, Arg_1 undercuts Arg_2 by showing a new premise which value conflicts with one of the premises of Arg_2 or else, appears in the description of the problem to solve but not in the problem description of Arg_2. Finally, in case 3 Arg_1 undercuts Arg_2 by putting forward a counter-example for a domain-case or an argument-case that was used to generate the conclusion of Arg_2.

Alternatively, the ASPIC framework considers a third kind of attack, the *undermining attack* [35, Definition 3.16], first presented in [17] and [47] as the *premise attack*. In our framework, this attack is represented by the undercutting attack of type 2, raised by putting forward a distinguishing premise.

4.4 The Notion of Defeat between arguments

Once possible conflicts between arguments have been defined, the next step in the formal specification of an AF is to define the defeat relation between a pair of arguments. This comparison must not be misunderstood as a strategical function to determine with which argument an argumentation dialogue can be won [37]. A function like this must also consider other factors, such as other arguments put forward in the dialogue or agents' profiles. Therefore, it only tells us something about the relation between two arguments.

The dependency relation over roles $Dependency_{S_t}$ and the agent's preference relation over values $Valpref_{ag_i}$ defined in our framework establish an *argument ordering* that is used to determine which attacks result in defeats. Thus, the argument ordering of the cased-based framework for agent societies proposed in this work is based on pre-defined relations over roles and on agents' preferences over values instead of on strict and defeasible rules, as proposed in [35, Definition 3.10]. Hence, the relation of defeat between two arguments is defined in our AF as follows.

Let $Arg_1 = \{\phi_1, value_1, < S_1 >\}$ posed by agent ag_1 and $Arg_2 = \{\phi_2, value_2, < S_2 >\}$ posed by agent ag_2 be two conflicting arguments and $Valpref_{ag_i} :<^{S_t}_{ag_i} \subseteq V \times V$, defines a reflexive, symmetric and transitive partial order relation over the values of the agent ag_i in the society S_t. Then:

Definition 4.8 (Defeat)**.** *Arg_1 defeats Arg_2*
if ((rebuts(Arg_1,Arg_2) $\wedge \sim$ undercut(Arg_2,Arg_1)) $\vee$ undercuts(Arg_1,Arg_2)) $\wedge$ ($value_1 <^{S_t}_{ag_1} value_2 \notin Valpref_{ag_1}$) $\wedge$ (Role(ag_1) $<^{S_t}_{Pow}$ Role(ag_2) $\notin Dependency_{S_t}$ $\wedge$ Role(ag_1) $<^{S_t}_{Auth}$ Role(ag_2) $\notin Dependency_{S_t}$)

Therefore, we express that the argument Arg_1 $defeats_{ag_1}$ from the ag_1 point of view the argument Arg_2 as $defeats_{ag_1}(Arg_1, Arg_2)$ if Arg_1 rebuts Arg_2 and Arg_2 does not undercut Arg_1 or else Arg_1 undercuts Arg_2, and these attacks succeed. That occurs if ag_1 does not prefer the value promoted by Arg_2 to the value promoted by Arg_1 and ag_2 does not have a power or authority relation with ag_1. The first type of defeat poses a stronger attack on an argument, directly attacking its conclusion. In addition, an argument can strictly defeat another argument if the first defeats the second and the second does not defeat the first.

Definition 4.9 (Strict Defeat)**.** *Arg_1 strictly defeats Arg_2 if Arg_1 defeats Arg_2 and Arg_2 does not defeat Arg_1*

4.5 The Acceptability Status of arguments

The acceptability status of arguments determines their status on the basis of their interaction. Only comparing pairs of arguments is not enough to decide if their conclusions are acceptable, since defeating arguments can also be defeated by other arguments. Taking into account the underlying domain theory of a dialectical system, arguments can be considered *acceptable*, *unacceptable* and *undecided* [16]. However, the acquisition of new information in further steps of the dialogue could change the acceptability status of arguments.

Therefore, to decide the acceptability status of arguments a proof theory that takes into account the dialogical nature of the argumentation process is necessary. To evaluate the acceptability of arguments by using a dialogue game is a common approach [37]. Dialogue games are interactions between two or more players, where each one moves by posing statements in accordance with a set or predefined rules [29]. In our AF, the acceptability status of arguments is decided by using a dialogue game and storing in the argument-case associated to each argument its acceptability status when the dialogue ends. However, the definition of this dialogue game is out of the scope of this report and is presented in [19, Chapter 4].

5 Example Scenario

To exemplify the use of our AF, let us propose a scenario of an open MAS that represents a water market society [10], where agents are users of a river basin (RB) that can buy or sell their water-rights to other agents. A water-right is a contract with the basin administration organism that specifies the rights that a user has over the water of the basin (e.g. the maximum volume that he can spend, the price that he must pay for the water or the district where it is settled[i]).

The framework can be easily extended to work with farmers that belong to different groups, representing farmer cooperatives (which group together farmers that grow the same products) or irrigation cooperatives (which group together farmers that use the same irrigation channel). In addition, agents can play different roles in each group and even act as representatives of a group. Thus, this is a complex scenario that requires an AF that is able to take into account the social context of agents to properly manage the argumentation process.

For clarity purposes, in this example all agents belong to the same group (the river basin RB). In this setting, suppose that two agents that play the role of farmers, F1 and F2, are arguing with a basin administrator, BA, to decide over a water-right transfer agreement that will grant an offered water-right of a farmer F3 to another farmer. Figure 10 shows a graphical representation of this scenario.

The behaviour of RB is controlled by a certain set of norms N_{RB}, its value preference order promotes economy (EC) over solidarity (SO) (promotes saving money in each transfer over being supportive with the personal needs of the basin users, SO<EC) and commands a dependency relation of charity (C) between two farmers (Farmer $<_c^{S_t}$ Farmer) and power relation (P) between a basin administrator and a farmer (Farmer $<_p^{S_t}$ BasinAdministrator). F1 prefers economy over solidarity (SO<EC) and has a set of norms N_{F1}, F2 prefers solidarity over economy (EC<SO) and has a set of norms N_{F2} and by default, BA has the value preference order of the basin (SO<EC) and a set of norms N_{BA}. Also, all agents have their own knowledge resources (domain-cases case-base, argument-cases case-base and argumentation schemes ontology).

The premises of the domain context would store data about the water-right transfer offer and other domain-dependent data about the current problem. For instance, the premises of the original problem could be as shown in Figure 11 and represent the identifier of the water right owner (*owner*), the offered volume in liters of water (*volume*), the price in euros per liter of water (*price*), the district where the water right is settled (*district*) and the area of this district in acres (*area*).

In the first step of the argumentation process, both farmers F1 and F2 will search their case-bases

[i]Following the Spanish Water Law, a water-right is always associated to a district.

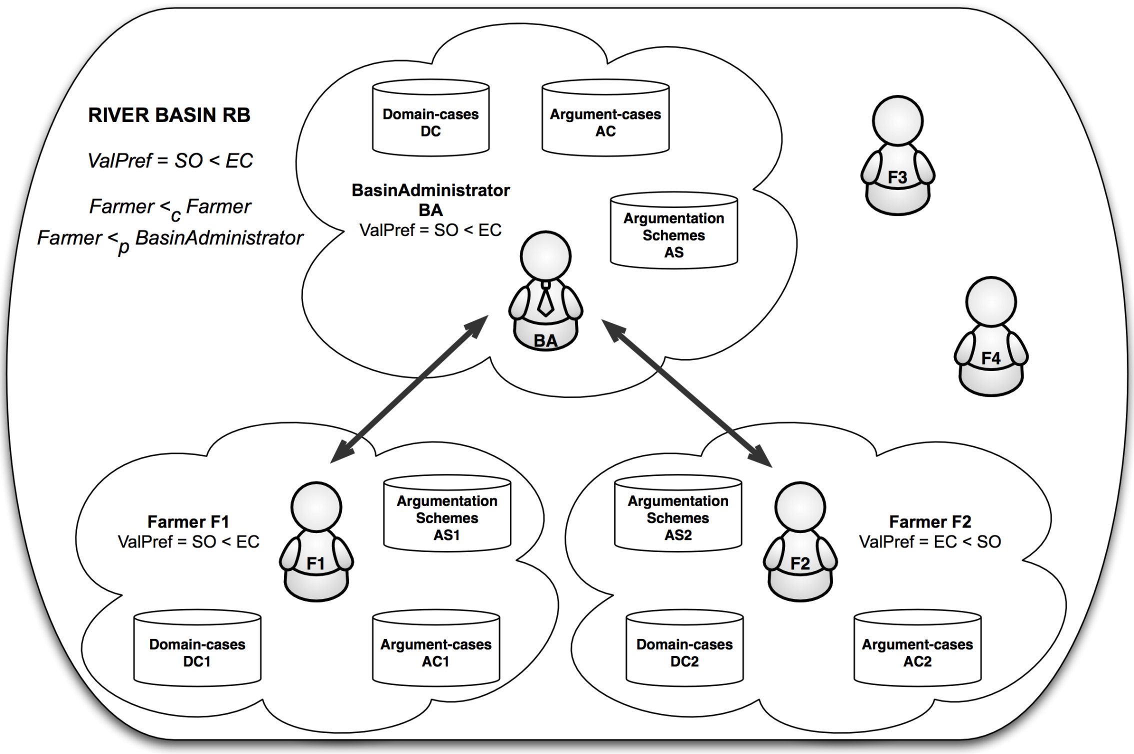


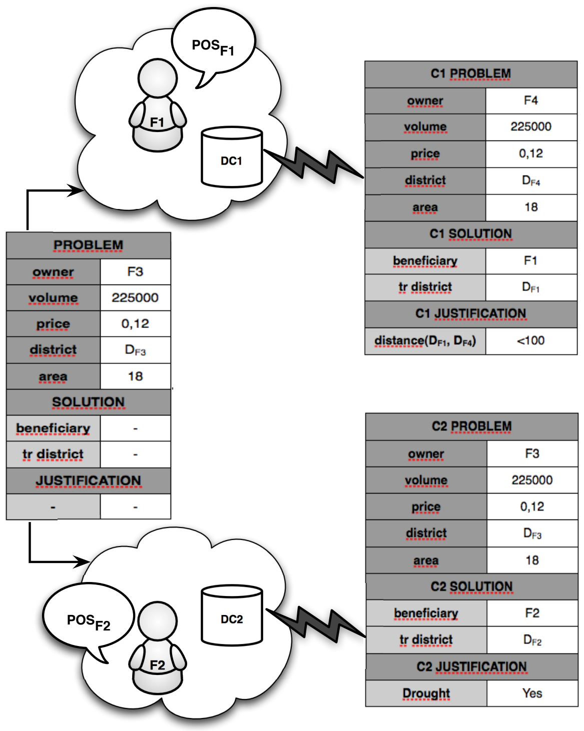
Figure 10: Water Market Scenario

of domain-cases (DC1 and DC2 respectively) to generate their positions. To query the case-bases, the problem is formatted as a target case without solution and justification, as shown in the left side of Figure 11. In this case, the solution consists of the identifier of the water-right transfer beneficiary (*beneficiary*) and the district of his land where the water has to be transferred (*tr district*). Figure 11 also shows how F1 has found a similar domain-case C1 that represents a similar water-right transfer that was granted to F1 since its land D_{F1} was adjacent (was closer than 100 meters) to the land where the water-right was offered. Therefore, F1 can generate position pos_{F1} that is on the side of F1[j].

In the case of F2, the figure shows that it has retrieved also a similar domain-case C2, which shows how the same water-right transfer was granted to F2 to irrigate its dry land during a drought. Therefore, F2 can generate a position that is on its side, pos_{F2}.

Once the agents have proposed their positions, the basin administrator BA has to decide between

[j]In this example we assume that agents only propose such positions that are on their side.



PROBLEM	
owner	F3
volume	225000
price	0,12
district	D_{F3}
area	18
SOLUTION	
beneficiary	-
tr district	-
JUSTIFICATION	
-	-

C1 PROBLEM	
owner	F4
volume	225000
price	0,12
district	D_{F4}
area	18
C1 SOLUTION	
beneficiary	F1
tr district	D_{F1}
C1 JUSTIFICATION	
distance(D_{F1}, D_{F4})	<100

C2 PROBLEM	
owner	F3
volume	225000
price	0,12
district	D_{F3}
area	18
C2 SOLUTION	
beneficiary	F2
tr district	D_{F2}
C2 JUSTIFICATION	
Drought	Yes

Figure 11: Generation of Positions

them. Therefore, it asks F1 and F2 to provide an argument for supporting their positions. Assuming that F1 and F2 are willing to collaborate, they can put forward the following arguments (according with the format proposed in Definition 4.3):

Support argument of F1:

$SAF1 == \{F1tr, EC, < Premises, \{C1\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >\}$

Support argument of F2:

$SAF2 = \{F2tr, SO, < Premises, \{C2\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >\}$

where the support set includes the *premises* of the problem description and the domain-cases used by F1 (C1) and F2 (C2) to generate their positions. F1tr and F2tr means that the transfer is granted to F1 and F2 respectively. According to the values of the agents, we suppose that the closer the lands the cheaper the transfers between them and then, SAF1 would promote economy (EC). Also, we assume that crops are lost in dry lands and helping people to avoid losing crops promotes solidarity (SO). Thus, SAF2 would promote solidarity.

Now, the BA has to attack the arguments of F1 and F2 and decide the beneficiary of the water-right transfer. Then, suppose that as basin administrator it knows an extra premise that states that there is a drought in the basin. First, this new premise matches an argumentation schemes of its ontology, S1, which changes the value preference order of the basin in case of drought (inspired by Waltons's *argument for an exceptional case* [48]):

> **Major Premise:** if the case of x is an exception, then the value preference order of the basin can be waived and changed by EC<SO in the case of x.
>
> **Minor Premise:** the case of drought is an exception.
>
> **Conclusion:** therefore the value preference order of the basin can be waived and changed by EC<SO in the case of drought.

Thus, this scheme will change the social context of the attack argument that the BA is going to create. As the support set of SAF1 and SAF2 contains a domain-case, the BA will try to propose a counter-example or a distinguishing premise for these cases.

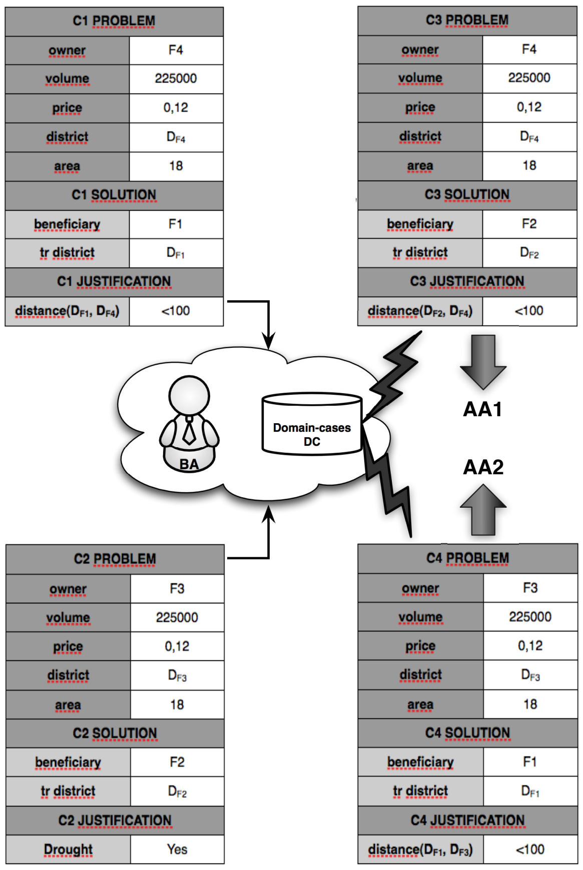
Thus, the BA will check its case-base of domain-cases (DC) to find counter-examples for C1 and C2. As shown in Figure 12, suppose that the BA finds one counter-example for each case (C3 for C1 and C4 for C2). Thus, it could generate the following attack arguments:

AA1 = {∼C1, SO, <Premises ∪ {Drought}, ∅, ∅, S1, ∅, ∅, ∅, {C3}>}

to undercut SAF1 by attacking its support element C1 with the counter-example C3. Here we assume that by attacking the argument of F1, the BA supports the argument of F2 and then promotes solidarity (SO).

AA2 = {∼C2, EC, <Premises ∪ {Drought}, ∅, ∅, S1, ∅, ∅, ∅, {C4}>}

to undercut SAF2 by attacking its support element C2 with the counter-example C4. Here we assume that by attacking the argument of F2, the BA supports the argument of F1 and then promotes economy (EC).



C1 PROBLEM	
owner	F4
volume	225000
price	0,12
district	D_{F4}
area	18
C1 SOLUTION	
beneficiary	F1
tr district	D_{F1}
C1 JUSTIFICATION	
distance(D_{F1}, D_{F4})	<100

C3 PROBLEM	
owner	F4
volume	225000
price	0,12
district	D_{F4}
area	18
C3 SOLUTION	
beneficiary	F2
tr district	D_{F2}
C3 JUSTIFICATION	
distance(D_{F2}, D_{F4})	<100

C2 PROBLEM	
owner	F3
volume	225000
price	0,12
district	D_{F3}
area	18
C2 SOLUTION	
beneficiary	F2
tr district	D_{F2}
C2 JUSTIFICATION	
Drought	Yes

C4 PROBLEM	
owner	F3
volume	225000
price	0,12
district	D_{F3}
area	18
C4 SOLUTION	
beneficiary	F1
tr district	D_{F1}
C4 JUSTIFICATION	
distance(D_{F1}, D_{F3})	<100

Figure 12: Counter-examples for C1 and C2

Then, it will try to find distinguishing premises and will check that the problem description of domain-cases C1 and C2 matches the extended description of the problem (the original description plus the new premise drought). Then, the BA realises that C1 does not match with the extended description and generates an attack argument to F1:

$$AA3 = \{\sim C1, SO, <Premises \cup \{Drought\}, \emptyset, \emptyset, S1, \emptyset, \{Drought\}, \emptyset, \emptyset >\}$$

to undercut SAF1 by attacking its support element C1 with the distinguishing premise *drought*. Again, we assume that attacking the argument of F1, the BA supports the argument of F2 and then promotes solidarity (SO).

Now, the BA has to select the argument that it will pose to attack the positions of the farmers. Note that, if we assume that agents always observe their value preference orders to put forward arguments,

PROBLEM	Domain Context	Premises = {owner=F3, volume=225000, ... drought=yes}	
	Social Context	Proponent	ID = BA
			Role = Basin Administrator
			Norms = N_{BA}
			ValPref = EC<SO
		Opponent	ID = F1
			Role = Farmer
			Norms = N_{F1}
			ValPref = SO<EC
		Group	ID = G
			Role = River Basin
			Norms = N_G
			ValPref = SO<EC
		Dependency Relation = Power	
SOLUTION	Argument Type = Inductive		
	Conclusion = F2tr		
	Value = SO		
	Acceptability Status = Unaccepted		
	Received Attacks	Critical Questions = ∅	
		Distinguishing Premises = ∅	
		Counter Examples = ∅	
JUSTIFICATION	Cases = {C5, C7}		
	Argumentation Schemes = ∅		
	Associated Dialogue Graph		

Table 2: Argument-case similar to AA3

the BA would prefer to pose AA1 and AA3 first than AA2 (since the BA has the value preference order of the basin, which has been changed to EC<SO). However, it has still to decide which AA1 or AA3 would select to attack SAF1. To do that, it generates an argument-case for each argument and checks its argument-cases case-base to decide which is the best argument to pose in view of the previous experience. Now, let us suppose that the BA finds a similar argument-case for AA3 that was unaccepted at the end of the dialogue, shown in Table 2. However, the information of the group that the agents belong does not match with the current one. Therefore, the BA can infer that in the argument represented by this argument-case the agents belonged to a different river basin where, solidarity is not promoted in case of drought. Finally, the BA finds a similar argument-case for AA1 that was accepted in the past. In this case, the social context and the value promoted match the current one. Thus, the BA will pose AA1 to attack the position of F1.

When F1 receives the attack, it has to evaluate the attack argument in view of its support argument. Then, it will realise that SAF1 does not defeat AA1 from its point of view, since the BA has a power dependency relation with any farmer ($Farmer <^{S_t}_{Power}$ $Basin\ Administrator$). Then, it would try to generate more support for its position. In case that it cannot find more support, the F1 would have to withdraw $pos_F 1$ and F2 would be granted the water-right transfer agreement.

With this scenario we have demonstrated how agents' arguments can be computationally managed in the proposed argumentation framework. The example shows the way in which agents can use the knowledge resources of the framework to generate, select and evaluate positions and arguments. Also, it takes into account the social context of agents to perform these activities and meets the requirements identified in Section 2. In [19, Chapter 6] our argumentation framework has been also implemented and tested to enhance a real customer support application run by a Spanish company [22].

On one hand, the framework is completely case-based, which makes it computationally tractable and eases the performance of automatic reasoning processes over it, taking advantage of previous argumentation experiences. On the other hand, the framework allows representing domain-dependent information in the premises and social information about the agents and their group. This information is used to select the best positions and arguments to put forward in each step of the argumentation process. In addition, the framework allows generating arguments from different knowledge resources and represents different types of arguments, supporting positions or attacking other arguments.

6 Discussion

This report presents a case-based argumentation framework for agent societies, where a generic KI case-based structure for computational argument representation has been proposed. As pointed out in Section 2.2, in open MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. However, tracking the arguments that agents put forward in argumentation processes could be relatively simple. Therefore, these arguments can be stored as cases codified in an ontological language that different agents are able to understand. This is easier than using a rule-based system, such as the ASPIC argumentation framework [35], which requires the creation of an explicit model of the domain. Nevertheless, the connections between the elements of our framework and their equivalences in the ASPIC

ArgCBROnto Concepts	**AIF Concepts**
<Premise ID, Premise Content>	<Premise Description, I-Node Premise>
Argument Type	Rule of Inference Scheme
Conclusion	<Conclusion Description, I-Node Conclusion>
Presumptions	<Presumption Description, I-Node Premise>
Exceptions	<ConflictScheme, Premise Description, I-Node Premise>
Distinguishing Premise	<Conflict Scheme, Premise Description, I-Node Premise>
Counter-Examples	<Conflict Scheme, Premise Description, I-Node Premise>

Table 3: Correspondence between ArgCBROnto and AIF concepts

framework have been discussed in Section 4.

Other important problem with rule-based systems arises when the knowledge-base must be updated (e.g. adding new knowledge that can invalidate the validity of a rule). Updates imply to check the knowledge-base for conflicting or redundant rules. Case-based systems are easier to maintain than rule-based systems since, in the worst case, the addition of new cases can give rise to updates in some previous cases, but does affect the correct operation of the system, although it can have an impact in its performance.

The representation of argumentation information using ontologies was first proposed in [52]. This work develops a format for argument interchange (AIF) that can be used between argumentation tools and/or MAS. The ArgCBROnto presented in this report follows the AIF approach, but defining a specific language for case representation that facilitates case-based reasoning processes over domain and argumentation information. However, an argumentation system based on our framework can interact with other systems that comply with the standard. Moreover, elements of cases are specified by using an ontological case representation language, the ArgCBROnto ontology. Although agents in open MAS are heterogeneous, by only sharing this ontology they can *understand* the arguments interchanged in the system. An example of the translation between some concepts of ArgCBROnto and their equivalent concepts in AIF (in the version reported in [40]) is shown in table 3.

AIF represents actual arguments as graphs with interlinked nodes that stand for the different concepts of the AIF ontology [52] [40]. For instance, premises in AIF are specified by using a description, which stands for the scheme that matches that premise and a description content, which is the actual information represented in the premise. Similarly, in ArgCBROnto premises have a name, which could be translated into the AIF description concept and a content, which corresponds to the AIF I-Node with the content of the premise. The same holds for presumptions and conclusions. The ArgCBROnto argument type has the same functionality as the AIF *rule of inference scheme*, which is the application of a specific type of inference (inductive, presumptive or mixed in our framework). In the case of exceptions, AIF represents them as relations between two concepts of the ontology interlinked via a special type of node called *conflict scheme*. This can be translated in the ArgCBROnto concepts of exceptions, distinguishing premises and counter-examples, which are represented by a a name, which again, could be translated into the AIF description concept and a content, which corresponds to the AIF I-Node with the content of the specific concept.

As pointed out in Section 1, in our case-based argumentation framework for agent societies we endorse the view of value-based argumentation frameworks [6], which stress the importance of the audience and the values promoted by an argument in determining whether it is persuasive or not. A related work on abstract argumentation scheme frameworks [5] combines argumentation frameworks with argumentation schemes and makes use of the structure provided by the schemes to guide dialogues and provide contextual elements of argument evaluation. However, these and most works on computational models of argument take a narrow view on the argument structure [39]. In fact, they are abstract frameworks aimed at studying the properties of arguments [45], which enable evaluation with respect to the logical relations between arguments. Opposite to our framework, the actual structure of arguments and their computational representation are obviated. In addition, previous argumentation experiences are not used to guide current argumentation processes such as we propose.

Other works use domain-dependent structures for the computational representation of arguments. The few current approaches for case-based argumentation in MAS, which use cases as previous knowledge to manage arguments, suffer from this domain-dependency or centralise the case-based argumentation abilities in a mediator agent [21]. Close to the approaches on case-based argumentation is the research on experience-based argumentation using association rules, presented as the *PADUA* protocol in [49]. This work pools the opinions of several agents that have access to different datasets to predict the classification of a new example. In a subsequent research, the PADUA protocol has been extended to allow multi-agent dialogues by proposing the *PISA* protocol [50, 51]. Here, the authors tackle issues like the dynamic creation of a group, the selection of a group leader and intra-group consultation to suggest moves. As in our approach, in this research agents take profit from previous experiences to solve a new problem, but the knowledge gained from the argumentation process is not stored nor used to improve the agents argumentation skills. In addition, PISA and PADUA have been designed to solve classification problems and not any type of problem that can be characterised by a set of features, which is the target of our research. However, the extension of our framework to allow agents to dynamically create groups and argue about the best move to make as a group in each step of the dialogue still remains future work.

In addition, the application of argumentation to agent societies is a new area or research with few contributions to date. However, commonly the term *agent society* is used in the argumentation and AI literature as a synonym for an *agent organisation* or a *group of agents* that play specific roles, follow some interaction patterns and collaborate to reach global objectives. Many works in argumentation in MAS that refer to the term 'agent societies' follow this approach, which is not targeted to the study of the structure of agent societies and the underlying social dependencies between agents.

This is the case of [18], which points out some of the drawbacks of classical 'agent centered MAS'. To resolve these difficulties the authors propose a set of principles and an example methodology to design organisation centered multi-agent systems. Also, [30] uses multi-agent argumentation within the agents and artifacts meta-model to design an argumentation component based on Dung's preferred semantics [16]. This component manages arguments and provides a coordination service for argumentation processes in a MAS.

Other works are focused on the study of argumentation in social networks, with a focus on the network topology (or the structure of the group) rather than in the actual social dependencies between software agents or human users. An example of this type is the work presented in [32], which investigates how argumentation processes among a group of agents may affect the outcome of group judgments in prediction markets. Also, an example on how argumentation can enhance dialogues in social networks can be found in [20].

However, the influence of the agent group and the social dependencies between agents in the way agents can argue must be further investigated. For instance, an agent playing the role of employee could accept arguments from an agent playing the role of project manager that it would never accept in other situation, such as playing the role of general manager. In the same way, an agent representing a group of employees (playing the role of trade unionist) is not expected to behave in the same way when arguing with an agent playing the role of the employer's representative than it does when arguing as an individual employee.

Recent research presents a novel argumentation-based negotiation framework that allows agents to detect, manage and resolve conflicts related to their social influences in a distributed manner within a structured agent society [25, 26]. This work proposes a new argumentation scheme that captures how agents reason about influences within a structured society, a mechanism to use this scheme to identify a suitable set of social arguments, a language and a protocol to exchange these arguments and a decision making functionality to generate such dialogues. However, it defines the social context of agents with a set of roles that agents can play, a set of generic relationships over them and a set of weighted social commitments for each of the active relationships, with no mention to values nor preferences over them.

A major difference between this proposal and our argumentation framework is the main objective pursued. Here the focus is on solving conflicts regarding conflicting social commitments between agents, while our framework enables argumentation to solve a generic problem by using previous experiences, taking into account the social dependencies between agents but also their preferences over a set of values. Also, the authors do not specify the types of dependency relations that agents can have, leaving this concept as a generic relation. In our framework, argumentation experience is stored and reused to support agents in making decisions about the best argument to put forward in a specific situation. Agents belong to a society that impose dependency relations on them, so they are related via power, authorisation or charity dependencies. Thus, the specific dependency relation between a pair of agents plays an important role in deciding if an argument posed in a past argumentation dialogue can be still persuasive in a current situation (where, possibly, agents hold a different dependency relation). For the time being, we do not deal with conflicts on dependency relations between agents, but this is an interesting extension that opens the pathway to future work.

In real systems, some features of argument-cases could be unknown. For instance, the proponent of an argument obviously knows its value preferences, probably knows the preferences of its group but, in a real open MAS, it is unlikely to know the opponent's value preferences. However, the proponent could know the value preferences of the opponent's group or have some previous knowledge about the value preferences of similar agents playing the same role as the opponent. If agents belong to different groups, the group features could be unknown, but the proponent could use its experience with other agents of the opponent's group and infer them. In any case, the framework is flexible enough to work with this lack of knowledge, although the reliability of the conclusions drawn from previous experiences would be worse.

For simplicity purposes, in the example proposed in this report we have assumed that a proponent agent addresses its arguments to an opponent of its same group, having complete knowledge of the social context. However, either the proponent or the opponent's features could represent information about agents that act as representatives of a group and any agent can belong to different groups at the same time. In that case, other issue that this research leaves open is the problem of solving conflicts between the values inherited from the group (or from the different groups of the agent) and the agent's individual values. The decision about which values are preferred would depend on the application domain. For instance, if the argumentation framework is implemented in a collaborative application domain where agents pursue to reach the best agreement for the whole group, the group values would be given priority over individual values.

Also for simplicity, the example does not show how agents can use the dialogue graphs associated to argument-cases to take strategic decisions about which arguments are more suitable in a specific situation or about whether continuing with a current argumentation dialogue is worth. For instance, to improve efficiency in a negotiation an argumentation dialogue could be finished if it is being similar to a previous one that didn't reach an agreement. Alternatively, opponent moves in a dialogue could be inferred by looking a similar previous dialogue with the same opponent.

The actual algorithms to implement the agents' reasoning process and the interaction protocol to interchange arguments between agents are not specified here. The former depends on the application domain and the design of the real system that implements our AF. The latter defines the dialogue, commitment and termination rules and the locutions that agents use to interchange arguments. These locutions depend on the agents' communication language and determine the intention of the argument (e.g. pose an attack or ask for justifications), the argument's sender and receiver, the format of the argument's content (e.g. if complete knowledge resources or parts are sent), etc. Also, the process to put critical questions depends on the actual ontology of argumentation schemes that agents implement and the interaction protocol that agents follow.

When we identified the research challenges for a case-based argumentation framework we noticed that due to the dynamism of the argumentation domain applied to open MAS, cases can quickly become obsolete. Therefore, there is an important opportunity here to investigate new methods for the maintenance of the case-bases that improve the adaptability of the framework. In this research, we have followed the basic approach to update cases when a new case that is similar enough to an existent case in the case-base has to be added. However, we acknowledge that this can give rise to too large databases with obsolete cases that can hinder the performance of the whole system.

7 Conclusions

The main contribution of this work consists in the proposal of a case-based argumentation framework that allows agents to argue in agent societies, taking into account their roles, preferences over values and dependency relations. Our notion of agent society is presented and a set of requirements that a suitable argumentation framework for agent societies should met is identified. The main advantages that our framework contributes over other existent AFs deal with these requirements. These advantages are: 1) the ability to represent social information in arguments; 2) the possibility of automatically managing arguments in agent societies; 3) the improvement of the agents' argumentation skills; and 4) the easy interoperability with other frameworks that follow the argument and data interchange web standards.

After that, we propose a set of knowledge resources that agents can use to manage positions and arguments in a way that they are computable. Moreover, these knowledge resources allow agents to use previous argumentation experiences to enhance their argumentation skills. The knowledge resources that agents use and the arguments that agents can interchange are ontologically defined in OWL-DL by using an ontology called ArgCBROnto, which allow heterogeneous agents in an open MAS to understand these concepts by sharing the ontology.

The framework proposed is formalised by defining the notion of argument, the logical language used to represent arguments, the concept of conflict between arguments, the notion of defeat between arguments and the acceptability status of arguments at the end of the argumentation dialogue.

An example scenario that shows how our framework allows agents to argue in a society taking into account the requirements identified is also proposed. The example demonstrates how the framework presented in this report meets all the necessary requirements for a suitable computational argumentation framework for agent societies.

Appendix: Description Logics

Description Logics (DLs) are a family of formal knowledge representation languages that are used in AI for formal reasoning on the concepts of an application domain (terminological knowledge). In DL the knowledge base consist of a set of terminological axioms (or *TBox*) that contains sentences describing relations between concepts and a set of assertional axioms (or *ABox*) that describes the relations between individuals and concepts (where in the hierarchy of concepts the individuals belong). Thus, DL distinguishes between concepts, roles, which are properties of these concepts and individuals, which are instances of the concepts. Table 4 shows the syntax and interpretation of the DL definitions provided in this report. In the table, C and D are concepts, R is a role, a and b are individuals and n is a positive integer.

Description	**Example**	**Interpretation**
All concept names	$\top$	Top
Empty concept	$\bot$	Bottom
Intersection of concepts	$C \sqcap D$	C and D
Union of concepts	$C \sqcup D$	C or D
Negation of concepts	$\neg C$	not C
Concept inclusion	$C \sqsubseteq D$	All C are D
Universal restriction	$\forall R.C$	All concepts with the role R are in C
Minimal cardinality	$\geq nR$	At least n concepts have the role R
Range	$\top \sqsubseteq \forall R.C$	The range of the role R is C
Domain	$\top \sqsubseteq \forall R^{-}.C$	The domain of the role R is C

Table 4: DL Notation

Funding

This work is supported by the Spanish government grants [CONSOLIDER-INGENIO 2010 CSD2007-00022, TIN2008-04446, and TIN2009-13839-C03-01] and by the GVA project [PROMETEO 2008/051].

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] L. Amgoud. An argumentation-based model for reasoning about coalition structures. In *2nd International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-05*, pages 1–12. ACM Press, 2005.

[3] L. Amgoud, L. Bodensta, M. Caminada, P. McBurney, S. Parsons, H. Prakken, J. van Veenen, and G. Vreeswijk. Project N 002307 ASPIC, Argumentation Service Platform with Integrated Components. Deliverable D2.6. Technical report, ASPIC Consortium, 15 Feb. 2006.

[4] A. Artikis, M. Sergot, and J. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1), 2009.

[5] K. Atkinson and T. Bench-Capon. Abstract argumentation scheme frameworks. In *Proceedings of the 13th International Conference on Artificial Intelligence: Methodology, Systems and Applications, AIMSA-08*, volume 5253 of *Lecture Notes in Artificial Intelligence*, pages 220–234. Springer, 2008.

[6] T. Bench-Capon and K. Atkinson. *Argumentation in Artificial Intelligence*, chapter Abstract argumentation and values, pages 45–64. Springer, 2009.

[7] T. Bench-Capon and P. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–938, 2007.

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

[9] T. J. M. Bench-Capon. Persuasion in Practical Argument Using Value-based Argumentation Frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[10] V. Botti, A. Garrido, A. Giret, and P. Noriega. Managing water demand as a regulated open MAS. In *Workshop on Coordination, Organization, Institutions and Norms in agent systems in on-line communities, COIN-09*, volume 494 of *LNCS*, pages 1–10. Springer, 2009.

[11] N. Bulling, J. Dix, and C. I. Chesñevar. Modelling coalitions: Atl + argumentation. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-08*, volume 2, pages 681–688. ACM Press, 2008.

[12] T. C. Bylander and B. Chandrasekaran. Generic tasks in knowledge-based reasoning: The right level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):231–243, 1987.

[13] B. Diaz-Agudo and P. A. Gonzalez-Calero. *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, volume 14 of *Integrated Series in Information Systems*, chapter An Ontological Approach to Develop Knowledge Intensive CBR Systems, pages 173–214. Springer, 2007.

[14] V. Dignum. *PhD Dissertation: A model for organizational interaction: based on agents, founded in logic.* PhD thesis, Proefschrift Universiteit Utrecht, 2003.

[15] V. Dignum and F. Dignum. A landscape of agent systems for the real world. Technical Report Technical Report 44-cs-2006-061, Intitute of Information and Computing Sciences, Utrecht University, 2006.

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

[17] M. Elvang-Gøransson, P. Krause, and J. Fox. Acceptability of arguments as 'logical uncertainty'. In *2nd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, ECSQARU-93*, pages 85–90. Springer-Verlag, 1993.

[18] J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In *Agent-Oriented Software Engineering VI*, volume 2935 of *LNCS*, pages 214–230. Springer-Verlag, 2004.

[19] S. Heras. *A Case-Based Argumentation Framework for Agent Societies.* PhD thesis, Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València, 2011.

[20] S. Heras, K. Atkinson, V. Botti, F. Grasso, V. Julián, and P. McBurney. How argumentation can enhance dialogues in social networks. In *Proceedings of the 3rd International Conference on Computational Models of Argument, COMMA-10*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 267–274. IOS Press, 2010.

[21] S. Heras, V. Botti, and V. Julián. Challenges for a CBR framework for argumentation in open MAS. *Knowledge Engineering Review*, 24(4):327–352, 2009.

[22] S. Heras, J. A. García-Pardo, R. Ramos-Garijo, A. Palomares, V. Botti, M. Rebollo, and V. Julián. Multi-domain case-based module for customer support. *Expert Systems with Applications*, 36(3):6866–6873, 2009.

[23] S. Heras, M. Navarro, V. Botti, and V. Julián. Applying Dialogue Games to Manage Recommendation in Social Networks. In *AAMAS 6th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-09*, pages 55–70. ACM Press, 2009.

[24] N. Karacapilidis and D. Papadias. Computer supported argumentation and collaborative decision-making: the HERMES system. *Information Systems*, 26(4):259–277, 2001.

[25] N. C. Karunatillake. *Argumentation-Based Negotiation in a Social Context.* PhD thesis, School of Electronics and Computer Science, University of Southampton, UK, 2006.

[26] N. C. Karunatillake, N. R. Jennings, I. Rahwan, and P. McBurney. Dialogue Games that Agents Play within a Society. *Artificial Intelligence*, 173(9-10):935–981, 2009.

[27] R. López de Mántaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M.-L. Maher, M. Cox, K. Forbus, M. Keane, and I. Watson. Retrieval, Reuse, Revision, and Retention in CBR. *The Knowledge Engineering Review*, 20(3):215–240, 2006.

[28] M. Luck and P. McBurney. Computing as interaction: agent and agreement technologies. In *IEEE International Conference on Distributed Human-Machine Systems.* IEEE Press, 2008.

[29] P. McBurney and S. Parsons. Dialogue games in multi-agent systems. *Informal Logic. Special Issue on Applications of Argumentation in Computer Science*, 22(3):257–274, 2002.

[30] E. Oliva, P. McBurney, and A. Omicini. Co-argumentation artifact for agent societies. In *5th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-08*, pages 31–46. ACM Press, 2008.

[31] S. Ontañón and E. Plaza. Learning and joint deliberation through argumentation in multi-agent systems. In *7th International Conference on Agents and Multi-Agent Systems, AAMAS-07.* ACM Press, 2007.

[32] S. Ontañón and E. Plaza. Argumentation-Based Information Exchange in Prediction Markets. In *Argumentation in Multi-Agent Systems*, volume 5384 of *LNAI*, pages 181–196. Springer, 2009.

[33] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[34] C. Perelman and L. Olbrechts-Tyteca. *The New Rhetoric: A Treatise on Argumentation.* University of Notre Dame Press, 1969.

[35] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, (1):93–124, 2010.

[36] H. Prakken, C. Reed, and D. Walton. Dialogues about the burden of proof. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law, ICAIL-05*, pages 115–124. ACM Press, 2005.

[37] H. Prakken and G. Sartor. A dialectical model of assesing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–368, 1996.

[38] I. Rahwan, F. Zablith, and C. Reed. Laying the foundations for a world wide argument web. *Artificial Intelligence*, 171(10-15):897–921, 2007.

[39] C. Reed and F. Grasso. Recent advances in computational models of natural argument. *International Journal of Intelligent Systems*, 22:1–15, 2007.

[40] G. Rowe and C. Reed. Diagramming the argument interchange format. In *2nd Conference on Computational Models of Argument, COMMA-08*, pages 348–359. IOS Press, 2008.

[41] J. Searle. *Rationality in Action.* MIT Press, 2001.

[42] D. Skalak and E. Rissland. Arguments and cases: An inevitable intertwining. *Artificial Intelligence and Law*, 1(1):3–44, 1992.

[43] L.-K. Soh and C. Tsatsoulis. A real-time negotiation model and a multi-agent sensor network implementation. *Autonomous Agents and Multi-Agent Systems*, 11(3):215–271, 2005.

[44] K. Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28:203–242, 1990.

[45] P. M. Thang, P. Dung, and N. D. Hung. Towards a Common Framework for Dialectical Proof Procedures in Abstract Argumentation. *Journal of Logic and Computation*, 19(6):1071–1109, 2009.

[46] P. Tolchinsky, K. Atkinson, P. McBurney, S. Modgil, and U. Cortés. Agents deliberating over action proposals using the ProCLAIM model. In *5th International Central and Eastern European Conference on Multi-Agent Systems and Applications, CEEMAS-07*, pages 32–41. Springer, 2007.

[47] G. A. W. Vreeswijk. *Studies in Defeasible Argumentation.* PhD thesis, Free University of Amsterdam, 1993.

[48] D. Walton, C. Reed, and F. Macagno. *Argumentation Schemes.* Cambridge University Press, 2008.

[49] M. Wardeh, T. Bench-Capon, and F. P. Coenen. PISA - Pooling Information from Several Agents: Multiplayer Argumentation From Experience. In *Proceedings of the 28th SGAI International Conference on Artificial Intelligence, AI-2008*, pages 133–146. Springer, 2008.

[50] M. Wardeh, T. Bench-Capon, and F. P. Coenen. Padua: a protocol for argumentation dialogue using association rules. *AI and Law*, 17(3):183–215, 2009.

[51] M. Wardeh, F. Coenen, and T. Bench-Capon. Arguing in groups. In *3rd International Conference on Computational Models of Argument, COMMA-10*, pages 475–486. IOS Press, 2010.

[52] S. Willmott, G. Vreeswijk, C. Chesñevar, M. South, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, and G. Simari. Towards an argument interchange format for Multi-Agent Systems. In *3rd International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-06*, pages 17–34. ACM Press, 2006.