Document downloaded from:

http://hdl.handle.net/10251/40614

This paper must be cited as:

Heras Barberá, SM.; Botti Navarro, VJ.; Julian Inglada, VJ. (2014). Modelling dialogues in agent societies. Engineering Applications of Artificial Intelligence. 34:208-226. doi:10.1016/j.engappai.2014.06.003.


The final publication is available at

http://dx.doi.org/10.1016/j.engappai.2014.06.003

Copyright International Federation of Automatic Control (IFAC)

# Modelling Dialogues in Agent Societies

Stella Heras, Vicente Botti, Vicente Julián

*Departamento de Sistemas Informaticos y Computacion*
*Universitat Politecnica de Valencia*
*Camino de Vera s/n 46022 Valencia (Spain)*

**Abstract**

Besides the simpler ability to interact, open multi-agent systems must include mechanisms for their agents to reach *agreements* by taking into account their social context. Argumentation provides multi-agent systems with a framework that assures a rational communication, which allows agents to reach agreements when conflicts of opinion arise. In this paper, we present the dialogue protocol that agents of a case-based argumentation framework can use to interact when they engage in argumentation dialogues. The syntax and semantics of the argumentation protocol are formalised and discussed. To illustrate our proposal, we have applied the protocol in the context of a water market. By using our dialogue protocol, agents represent water users that are able to explore different water allocations and justify their views about what is the best water distribution in a certain environment.

*Keywords:* Agreement Technologies, Argumentation, Multi-Agent Systems

## 1. Introduction

Large scale computer systems can be viewed in terms of the entities that participate in them, offering and consuming services (Luck and McBurney, 2008). Open Multi-Agent Systems (MAS), whose software agents are able to interact with each other to solve complex tasks and reach agreements as the outcome of their interactions, has proven to be a very appropriate paradigm to implement these type of systems (Huhns et al., 2005)(Ossowski, 2013)(del Val et al., 2014). Furthermore, argumentation theory provides MAS with a framework that assures rational communication and allows agents to reach agreements when conflicts of opinion arise. However, agents that use an argumentation framework to argue also need a protocol to communicate, to interchange their arguments, and to be able to reach agreements.

Considerable research has been performed on the design of artificial agent communication languages, such as the *Knowledge Query and Manipulation Language (KQML)*[1] from DARPA, and the *Agent Communications Language (FIPA ACL)*[2] from the IEEE Foundation for Intelligent Physical Agents. These languages provide agents with high flexibility of expression. However, in a dialogue, agents can have too many choices of what to utter in each step of the conversation. Therefore, this flexibility can also be an important downside if it gives rise to a state-space explosion and leads agents to engage in never-ending dialogues (McBurney and Parsons, 2009, Chapter 13).

A possible solution for this problem consists of limiting the allowed set of utterances for each step of the dialogue by defining the agent communication protocol by means of a *dialogue game* (Hamblin, 1970)(MacKenzie, 1979). Dialogue games are a concept from argumentation theory and game theory that has been applied in MAS to structure the dialogue between agents with different points of view. Formal dialogue games are interactions among several players (agents in our case) where each player moves by making utterances in accordance with a defined set of rules. A wide range of approaches that formalise interaction protocols by using different dialogue games have been published (McBurney and Parsons, 2002a).

However, to our knowledge no research has been done to propose a dialogue game that is based on case-based knowledge resources that agents can use to manage agreement processes in agent societies. Reasoning with cases is especially suitable where there is a weak (or even unknown) domain theory, but acquiring examples encountered

*Email address:* sheras@dsic.upv.es (Stella Heras, Vicente Botti, Vicente Julián)

[1] www.cs.umbc.edu/research/kqml/

[2] www.fipa.org/repository/aclspecs.html

in practice is easy. Many argumentation models for MAS produce arguments by applying a set of inference rules (Amgoud et al., 2000)(Augusto and Simari, 2001)(Verheij, 2009). Rule-based systems require eliciting an explicit model of the domain (Prakken, 2010). In open MAS, the domain is highly dynamic and the set of rules that model it is difficult to specify in advance, even if these rules are domain-specific inference rules that are intended to represent domain knowledge. However, tracking the arguments that agents put forward in argumentation processes can be relatively simple. Therefore, these arguments can be stored as cases that are codified in a specific case representation language that different agents are able to understand (e.g., an ontological language (Jurisica et al., 2004)). This approach makes possible to develop case-bases reducing the knowledge-acquisition bottleneck. With case-bases, agents are able to perform *lazy learning* processes on argumentation information. For complex and highly dynamic systems, this is easier than using a rule-based system.

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 involve checking 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 it does not affect the correct operation of the system, even though it can have an impact on its performance.

Therefore, in this paper, we present a dialogue game protocol that agents can use in a case-based argumentation framework to interact with each other when they engage in dialogues. This protocol includes a syntax as the set of defined locutions that agents can use to engage in argumentation processes, the combinatorial properties of locutions, and the rules that govern the dialogue. We also provide the *operational* semantics of the locutions. This semantics views each locution as a transition in an abstract state-machine that represents the possible stages that can be reached during the dialogue.

The structure of this paper is as follows: Section 2 introduces a running example that clarifies the type of problems that we want to solve with our argumentation approach; Section 3 briefly introduces our case-based argumentation framework for agent societies; Section 4 shows the syntax and operational semantics of the protocol and provides a discussion on its properties; Section 5 develops the running example in a dialogue among several agents in a water market that is controlled by our protocol; Section 6 analyses related work and compares it with our proposal; and Section 7 summarises the contents of this paper.

## 2. The Water Market Scenario

As in human societies, agents in agent societies have a social context that can impose on them a set of norms to obey, a preference order regarding a set of values that agents can promote with their actions, and a set of dependency relations that link them. By the mere fact of belonging to a group, an agent may have to comply with the norms of the group or to act in a way that promotes the values that the group prefers. Similarly, an agent that is under contract with another agent to provide it with a service is committed to accepting requests from the contracting party that it might never accept otherwise. To clarify this point, let us assume a real scenario where the social context of agents has a decisive influence on the agents' behaviour.

The example scenario consists of a water market where a society $S$ of agents that represent different users must reach an agreement over a water-right transfer. This scenario was introduced in the *mWater* prototype (Botti et al., 2009b)(Botti et al., 2009a)(Botti et al., 2010)(Garrido et al., 2009). Fresh water will be the "gold" of the 21st century (Honey-Roses, 2007). Only 3% of the Earth's water is salt free. Of that 3%, approximately 2.7% is frozen in polar ice caps or deep underground. This leaves only 0.3% of all the water on the planet available for human use (Schneider, 1996). Water scarcity is especially problematic in dry climates such as the Mediterranean. Spain already suffers from severe water shortages (Honey-Roses, 2007)(Panayotou, 2007). During the last few years, a dramatic change in the Spanish Water Law has given rise to many water problems. Spain needs to improve its water management in order to meet the needs of different types of users (e.g., farmers, cities, and private companies) and to deal with its severe water scarcity problems.

In this scenario, agents are users of a river basin that can buy or sell their water rights to other agents. A water right is a contract with the basin administration authority that specifies the rights that a user has over the water of the basin (e.g., the maximum volume that the user can use, the price that the user must pay for the water, or the district where the water right is located[3]). For instance, a particular water right could allow its holder to pump up to 10 $m^3$ of water per day during the next cotton season. It is possible to consider both the seller and the buyer as

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

grouped entities (instead of having only one member playing the role of seller/buyer, a set of members may join together to participate in the market on a larger scale). For instance, a given seller has a water right of 2 $m^3$ per day, which is clearly insufficient for a buyer that needs 10 $m^3$ of water. If more sellers are grouped together it would be possible to have water rights to fit the requirement of the buyer, which analogously can be grouped in a larger buyer entity. Now, the stakeholders of this scenario will need to take into consideration the seller/buyer entity and model the interactions among the particular members of each entity.

Our domain scenario assumes that several users are arguing to reach an agreement over a water-right transfer. In this scenario, agents can play the following roles (Giret et al., 2010):

- Water User: a water-right holder of the basin, for instance, a farmer.
- Buyer: a Water User that wants to transfer its right and or buy a transportation resource.
- Seller: a Water User that wants to purchase rights and or sell a transportation resource.
- Third party: a Water User that can be affected by a water-right transfer agreement.
- Basin regulating authority (Basin Administrator): the Basin Administration representative that can authorize a water-right transfer agreement.
- Jury: the referee entity for problems among the contracting parties and (possibly) third parties of a water-right transfer agreement.

Let us propose a concrete example for this scenario, where two agents that play the role of buyers and represent farmers ($F1$ and $F2$) in a group (the river basin $RB$) are arguing to decide over a water-right transfer agreement that will grant an offered water right of a farmer $F3$ playing the role of seller to another farmer. Figure 1 shows a graphical representation of this scenario.

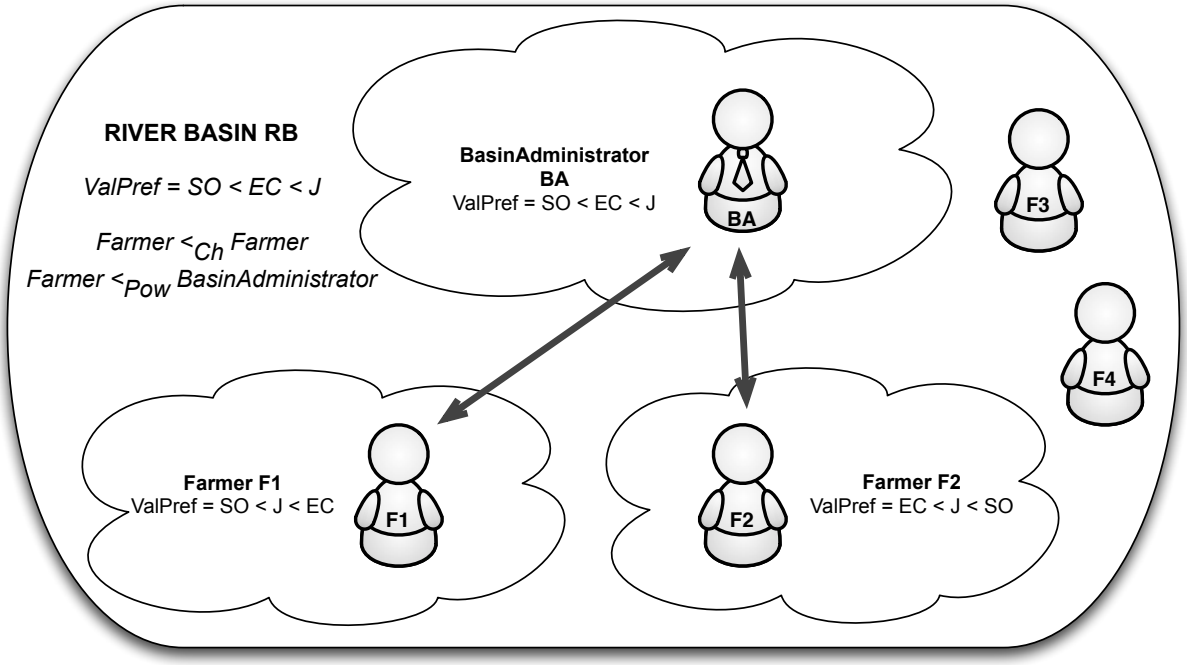


Figure 1: Water Market Scenario

Here, a basin administrator ($BA$) controls the process and makes a final decision. The behaviour of the basin is controlled by a certain set of norms $N_{RB}$. The society commands a charity (Ch) dependency relation between two water users (farmers) ($Farmer <_{Ch} Farmer$) and a power (Pow) dependency relation between an administrator (basin administrator), and a buyer (farmer) ($Farmer <_{Pow} BasinAdministrator$). A power relation of an agent over

another agent establishes a hierarchy for them, committing the second agent to accept the orders and requests of the first agent. A charity relation establishes a relationship of equality between two agents. Farmers usually prefer to reach an agreement before taking legal action in order to avoid the intervention of a jury (J). Also, $F1$ prefers to improve its economy (EC) over the intervention of a jury and this intervention over promoting the solidarity between users (SO) ($SO < J < EC$). $F2$ prefers solidarity over the intervention of a jury and this over economy ($EC < J < SO$). By default, *BA* adopts the value preference order of the basin (which promotes saving money in each transfer over being supportive of the personal needs of the basin users) and tries to avoid the intervention of a jury in any case ($SO < EC < J$).

This is a complex scenario that requires an argumentation framework and an underlying dialogue protocol that is able to take into account the social context of agents to be able to properly manage the argumentation process. For instance, at a certain point of the argumentation dialogue, the BA could put forward an argument that attacks the argument that a farmer has brought up to justify that it should be the beneficiary of the water-right transfer. If the social context of these agents is not considered, obviously the farmer would try to rebut the attack. However, this would violate the norms of the basin, that commits the farmers to accept the arguments of the administrator, even if these arguments do not promote the farmers' preferences. Furthermore, the agent that acts as basin administrator could personally prefer the intervention of a jury in spite of taking the responsibility to make a final decision about who should be the beneficiary of the transfer. However, as the basin representative, it has to adopt the value preference order of the basin and put forward as many arguments as possible to avoid the intervention of a jury in the agreement process, which might increase the financial costs of the process. Therefore, agents need to be able to engage in an argumentation process in order to reach an agreement on the final beneficiary of the transfer.

## 3. Case-based Argumentation Framework

In the PhD work developed in (Heras, 2011), a case-based argumentation framework that takes into account the social context of agents was proposed. Our framework has been implemented as an argumentation API in the *Magentix2* agent platform, which provides new services and tools that allow for the secure and optimised management of open MAS (which is publicly available at *http://www.gti-ia.upv.es/sma/tools/magentix2/*). In this section, we briefly introduce the elements of this framework. Specifically, our framework consists of several knowledge resources that the agents can use to generate, select, and evaluate arguments following a reasoning process to perform these tasks as well as a dialogue protocol that allow agents to reach agreements by performing this argumentative reasoning, which is the focus of this paper. The knowledge resources proposed in the framework are:

**A database of argumentation schemes** with a set of argumentation schemes (Walton et al., 2008), 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 an associated 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. For instance, the water-right transfer domain could include a scheme that represents a common pattern of reasoning in the agent society $S_t$ that the basin administrator follows and that changes the value preference order of the basin in case of drought (inspired in Waltons's *argument for an exceptional case* (Walton et al., 2008)):

> **Major Premise:** If the case of x is an exception, then the value preference order of the basin can be waived and changed by $EC <_{RB}^{S_t} J <_{RB}^{S_t} SO$ 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 $EC <_{RB}^{S_t} J <_{RB}^{S_t} SO$ in the case of drought.

**A case-base with domain-cases** that represent previous problems and their solutions. Agents can use this knowledge resource to generate their positions in a dialogue and arguments to support them. Also, the acquisition of new domain-cases increases the knowledge of agents about the domain under discussion. The domain case-base of the farmers in our example will store information about previous water-right transfer processes and their outcome (who the beneficiary was and under what terms). For instance, let us assume that a farmer agent $F2$ is granted a water-right transfer from its original owner $F3$ to promote solidarity, since it needs an

Table 1: Domain-Case $C2$

| | | |
|---|---|---|
| **PROBLEM** | Owner | F3 |
| | Volume | 225000 |
| | Price | 0.12 |
| | District | $D_{F3}$ |
| | Area | 18 |
| | Drought | Yes |
| **SOLUTION** | Beneficiary | F2 |
| | Transferred District | $D_{F2}$ |
| | Value Promoted | SO |
| **JUSTIFICATION** | Emergency | Drought |

urgent irrigation of its land during a drought. The volume of water transferred is of 225.000 liters at a price of 0.12 Euros per liter and $D_{F3}$ has an area of 18 acres. Therefore, $F2$ will store in its case-base the domain-case $C2$ (shown in Table 1) to represent the knowledge gained from this interaction[4].

**A case-base with argument-cases** that store previous argumentation experiences and their final outcome. Argument-cases have three main objectives: they can be used by agents 1) to generate new arguments; 2) to strategically select the best position to put forward in view of past argumentation experiences; and 3) to store the new argumentation knowledge gained in each agreement process, improving the agents' argumentation skills. The case-base of argument-cases of the farmers of the water-right transfer scenario will store information about the arguments that these farmers put forward to be selected as beneficiaries of the transfer in previous agreement processes. For instance, let us assume that in a new dialogue with the basin administrator of a group $G$, the argument of $F2$ supporting its candidacy as beneficiary of the transfer $F2tr$ was rejected (for instance, since in this river basin economic values prevail over solidarity and the administrator prefers to authorize the transfer to another irrigator). Thus, the farmer agent $F2$ would store in its case-base an argument-case representing the knowledge that it has gained about this transaction (see Table 2 for an example).

We use ontologies as the representation language for the knowledge resources of our framework. Specifically, we assume that domain-cases are instances of a domain-dependent ontology. Argumentation schemes are represented by using the *Argument Interchange Format (AIF)* ontology, as proposed in (Rahwan et al., 2011). Also, to represent argument-cases, we have created a case-based argumentation ontology, called *ArgCBROnto*[5].

The structure of domain-cases and the specific set of argumentation schemes that an argumentation system that implements our framework has depends on the application domain. Argument-cases are the main structure that we use to computationally represent arguments in agent societies. In addition, their structure is generic and domain-independent. Therefore, in this section, we focus on explaining the argument-case structure. Argument-cases have the same three possible types of components that usual cases of CBR systems have: 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*). Figure 2 shows the generic structure of an argument-case.

The problem description has a *domain context* that consists of the *premises* that characterise the argument. In addition, if we want to store an argument and use it to generate a persuasive argument in the future, the features that characterise its *social context* must also be kept. The social context of the argument-case includes information about the *proponent* and the *opponent* of the argument and about their *group*. Moreover, we also store the preferences (*ValPref*) of each agent or group over the set of *values* that are pre-defined in the system. Finally, the *dependency relation* between the proponent's and the opponent's roles is also stored. In our framework, we consider three types of dependency relations as defined in (Dignum and Weigand, 1995): *Power*, when an agent has to accept a request from another agent because of some pre-defined domination relationship between them; *Authorisation*, when an agent has signed a contract with another agent to provide it with a service and hence, the contractor agent is able to impose its authority over the contracted agent, and *Charity*, when an agent is willing

[4]This is based on the example developed in section 5

[5]The complete specification of the ArgCBROnto ontology can be found at:
`http://gti-ia.dsic.upv.es/~vinglada/docs/Sitio_web/ArgCBROnto.html`.

Table 2: Argument-case example

| | | | |
|---|---|---|---|
| **PROBLEM** | Domain Context | Premises = {owner=F3, volume=225000, .... drought=yes} | |
| | Social Context | Proponent | ID = BA |
| | | | Role = Basin Administrator |
| | | | Norms = $N_{BA}$ |
| | | | ValPref = $SO <^{S_t}_{BA} EC <^{S_t}_{BA} J$ |
| | | Opponent | ID = F2 |
| | | | Role = Farmer |
| | | | Norms = $N_{F1}$ |
| | | | ValPref = $EC <^{S_t}_{F1} J <^{S_t}_{F1} SO$ |
| | | Group | ID = G |
| | | | Role = River Basin |
| | | | Norms = $N_G$ |
| | | | ValPref = $SO <^{S_t}_{BA} EC <^{S_t}_{BA} J$ |
| | | Dependency Relation = Power | |
| **SOLUTION** | Argument Type = Inductive | | |
| | Conclusion = F2tr | | |
| | Value = SO | | |
| | Acceptability State = Unaccepted | | |
| | Received Attacks | Critical Questions = $\emptyset$ | |
| | | Distinguishing Premises = $\emptyset$ | |
| | | Counter Examples = $\emptyset$ | |
| **JUSTIFICATION** | Cases = {C2} | | |
| | Argumentation Schemes = $\emptyset$ | | |
| | Associated Dialogue Graphs | | |

to answer a request from another agent without being obliged to do so. For instance, as pointed out above, in the water-rights transfer scenario, the basin administrator has a power dependency relation over the farmers, while they have a charity relation with each other.

The *conclusion* of the case, the *value* promoted, and the *acceptability status* of the argument at the end of the dialogue are stored in the solution part. The acceptability status shows if the argument was deemed *acceptable*, *unacceptable*, or *undecided* in view of the other arguments that were put forward in the agreement process. In addition, the conclusion part includes information about the possible *attacks* that the argument received during the process. 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. Specifically, arguments in our framework can be attacked by putting forward *distinguishing premises* or *counter-examples* to them, as proposed in (Bench-Capon and Sartor, 2003), and also by questioning the validity of the conclusion drawn from an argumentation scheme by instantiating a *critical question*.

Let us assume that we have a set of cases denoted as $C$, a set of premises denoted as $F$, a problem to solve denoted as $P$ (characterised by a subset of the premises of $F$), and a function $value_c(x)$ that returns the value of a premise $x \in F$ in a case $c \in C$.

**Definition 3.1** (Distinguishing Premise)**.** *A distinguishing premise $x$ with respect to a problem $P$ between two cases $c_1, c_2 \in C$ is defined as: $\exists x \in c_1 \wedge \nexists x \in P \mid \exists x \in c_2 \wedge value_{c_1}(x) \neq value_{c_2}(x)$ or else, $\exists x \in c_1 \wedge \exists x \in Pmidvalue_{c_1}(x) = value_P(x) \wedge \nexists x \in c_2$, where $P \subseteq F$, $x \in F$ and $c_1, c_2 \in C$.*

Otherwise stated: a premise that does not appear in the description of the problem to solve and has different values for two cases or a premise that appears in the problem description and does not appear in one of the cases. For instance, in our example, if the problem specification does not include a premise that indicates that there is drought in the river basin, the premise *Drought* of $C2$ can be used by another agent to attack an argument of $F2$ that includes $C2$ as piece of evidence to support $F2$ position.

**Definition 3.2** (Counter-Example)**.** *A counter-example for a case $c_1 \in C$ with respect to a problem $P$ is another case $c_2 \in C$ such that: $acceptable(c_2) \wedge \forall x_i \in c_2 \cap P \mid value_{c_2}(x_i) = value_P(x_i) \wedge \forall x_i \in c_1 \mid (\exists x_i \in c_2 \wedge value_{x_i}(c_2) = value_{x_i}(c_1)) \wedge conclusion(c_2) \neq conclusion(c_1)$*

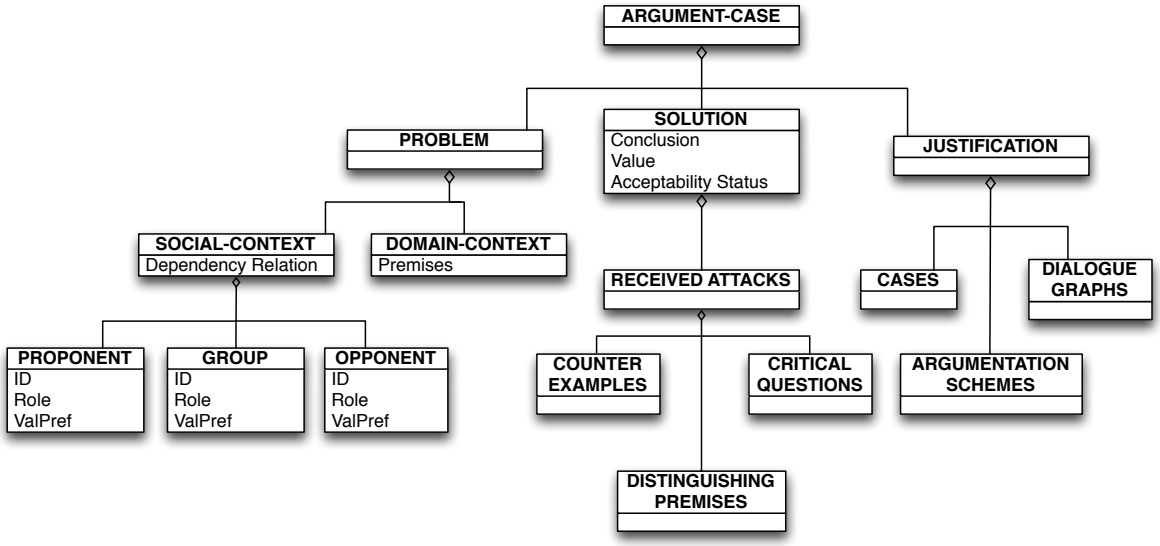


Figure 2: Structure of an Argument-Case

Otherwise stated: a counter-example for a case is a previous case (i.e., 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. In our example, a counter-example for $C2$ would be another domain-case that represents the situation where a similar transfer (in terms of quantity of water, price, land extension, area, etc.) was assigned to another beneficiary.

**Definition 3.3.** *A critical question is a question associated with an argumentation scheme that represents a potential way in which the conclusion drawn from the scheme can be attacked.*

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 (Prakken et al., 2005). In the case of presumptions, the proponent has the burden of proof if the critical question is asked, whereas in the case of the exceptions the burden of proof falls on the opponent that has questioned the conclusion of the scheme. Therefore, if the opponent asks a critical question, the argument that supports this argumentation scheme remains temporally rebutted until the question is conveniently answered. This characteristic of argumentation schemes makes them very suitable to devise ways to attack the conclusions drawn from other agents. For instance, in our example the argument-scheme presented in this section could include an exception to capture the fact that for a specific river basin, the case of drought is not considered as an exception. Therefore, if an agent can provide pieces of evidence to rise and justify this exception, the conclusion of the argument-case would be invalidated and the value preference order of the associated basin would remain unchanged.

Finally, the justification part of an argument-case stores the information about the knowledge resources that were used to generate the argument represented by the argument-case (the set of domain-cases, argument-cases, and argumentation schemes). In addition, the justification of each argument-case has an associated *dialogue-graph* (or several), which represents the dialogue where the argument was proposed. In this way, the sequence of arguments that were put forward in a dialogue is represented (storing the complete conversation as a directed graph that links argument-cases). This graph can be used later to improve the efficiency in an argumentation dialogue in view of a similar dialogue that was held in the past.

As pointed out above, in our framework, agents can generate arguments from previous cases (domain-cases and argument-cases) and from argumentation schemes. However, 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 among agents is not the same. Thus, arguments that agents interchange are defined as tuples of the form:

**Definition 3.4** (Argument). *$Arg = \{\phi, v, \{S\}\}$, where $\phi$ 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).*

This support set can consist of different elements, depending on the purpose of the argument. 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 proposed (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 also been 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*). Also, a supporting argument promotes the value promoted by the position that it justifies. 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*, *distinguishing premises*, or *counter-examples*). In our framework, we assume that an attack argument promotes the value promoted by the position that it tries to defend (if an agent has generated it to rebut an attack on its supporting argument) or otherwise, an attack argument promotes the agent's most preferred value over the set of values that is pre-defined in the system (if an agent has generated it to attack the position of other agent).

For instance, in the water-right transfer domain, $Arg = \{F2tr, SO, \{C2\}\}$, would represent the argument that farmer F2 has generated by using its domain-case C2 to justify that it should be the beneficiary of the transfer (F2tr) to save his crop in a drought emergency (promoting solidarity (SO)).

## 4. Dialogue Game Protocol

To formalise the protocol that agents use to engage in argumentation processes by using our framework, we follow a *dialogue game* approach. Dialogue games are interactions between two or more players, where each player 'moves' by making statements that follow a pre-defined set of rules (McBurney and Parsons, 2002a). Dialogue games are a specific type of games from *game theory* that are different from the classical games studied in Economics, in the sense that the profits or losses for the victory or defeat are not considered. Another important difference is that, in dialogue games, the participants are not able to model the potential moves of other participants by using an uncertainty measure, for instance, a probabilistic measure. These characteristics make dialogue games a methodology that is suitable for modeling the interactions among heterogeneous agents in a dynamic environment.

Specifically, we follow the dialogue game approach proposed in (McBurney and Parsons, 2002b) and extended in (McBurney and Parsons, 2009). This approach is prospective (intended to model systems in order to represent reality and that do not exist yet), which fits the objective of most open MAS. Other approaches for formalising dialogue systems have been reviewed in (Prakken, 2006) (specifically, formal systems for persuasion dialogue). However, most of these proposals are retrospective (intended to reconstruct/explain what happened in a dialogue, using a legal dispute as typical example). Furthermore, they assume a consistent and presupposed *context* that represents fixed and indisputable knowledge that cannot be changed during the dialogue. This assumption cannot be made in open MAS where heterogeneous agents with partial knowledge about the context of the dispute can enter or leave the system (and hence the dialogue) at any time.

Throughout this paper, we assume that a set of agents with different positions (points of view) are arguing to reach an agreement to solve a complex problem. Thus, our basic notion of agreement consists of a solution for a generic problem that several agents must solve. At this level of abstraction, we assume that this is a generic problem of any type (e.g., resource allocation, classification, prediction, etc.) that could be described with a set of features. However, different notions of agreement can be found in the literature of agreement technologies (Carrascosa and Rebollo, 2009). First, we introduce the notation that we use in defining the protocol. Subsequently, the protocol syntax and semantics are presented. Finally, we provide a discussion on the protocol properties.

### *4.1. Notation*

In our dialogue protocol we follow the standard that views utterances as composed by two layers: an internal layer that represents the *topics* of the dialogue and an external layer that consists of the *locutions* or performatives that define the allowed speech acts. On one hand, we assume that the topics of the inner layer can be represented with well-formed formulae of the Description Logic (DL) $\mathcal{SHOIN}(D)$ (Horrocks and Patel-Schneider, 2004), which forms the basis of the Web Ontology Language OWL-DL. As pointed out above, we have designed an ontology called *ArgCBROnto* to define the representation language of arguments and argumentation concepts. Ontologies provide a common vocabulary to understand the structure of information among different software

agents. In addition, ontologies allow assumptions about the domain to be made explicit, which facilitates to change these assumptions as new knowledge about the domain is acquired. The high dynamism of the domains where open MAS operate gives rise to many changes in the domain knowledge that agents have available. Therefore, they must be able to efficiently handle the consequences of these changes. On the other hand, we use the standard operators and axioms of *modal logics* of knowledge and belief (Shoham and Leyton-Brown, 2009, Chapter 13) to define the semantics of locutions.

In DLs, the important notions of the domain are described by *concept descriptions*, which are expressions that are built from atomic *concepts* (unary predicates) and atomic *roles* (binary predicates relating concepts) using the concept and role constructors provided by the specific DL. The semantics of DLs is given in terms of *interpretations* (Baader et al., 2007). Table 3 shows the syntax and semantics of the constructors of $\mathcal{SHOIN}(D)$, using Roman upper-case letters to represent *concepts*, *datatypes*, and *roles* and Roman lower-case letters to represent *individuals* and *data values*.

Table 3: Syntax and Semantics of $\mathcal{SHOIN}(D)$ (Horrocks and Patel-Schneider, 2004).

| **Constructor Name** | **Syntax** | **Semantics** |
|---|---|---|
| atomic concept **A** | $A$ | $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| datatypes **D** | $D$ | $D^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^{\mathcal{I}}$ |
| abstrac role $\mathbf{R}_A$ | $R$ | $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| datatype role $\mathbf{R}_D$ | $U$ | $U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathcal{D}}^{\mathcal{I}}$ |
| individuals **I** | $o$ | $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ |
| data values | $v$ | $v^{\mathcal{I}} = v^{\mathcal{D}}$ |
| inverse role | $R^{-}$ | $(R^{-})^{\mathcal{I}} = (R^{\mathcal{I}})^{-}$ |
| conjunction | $C_1 \sqcap C_2$ | $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ |
| disjunction | $C_1 \sqcup C_2$ | $(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$ |
| negation | $\neg C_1$ | $(\neg C_1)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C_1^{\mathcal{I}}$ |
| oneOf | $\{o_1, ...\}$ | $\{o_1, ...\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, ...\}$ |
| exists restriction | $\exists R.C$ | $(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\, \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$ |
| value restriction | $\forall R.C$ | $(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\, \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$ |
| atleast restriction | $\geq nR$ | $(\geq nR)^{\mathcal{I}} = \{x \mid \#(\{y.\, \langle x, y\rangle \in R^{\mathcal{I}}\}) \geq n\}$ |
| atmost restriction | $\leq nR$ | $(\leq nR)^{\mathcal{I}} = \{x \mid \#(\{y.\, \langle x, y\rangle \in R^{\mathcal{I}}\}) \leq n\}$ |
| datatype exists | $\exists U.D$ | $(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y.\, \langle x, y\rangle \in U^{\mathcal{I}} \text{ and } y \in D^{\mathcal{D}}\}$ |
| datatype value | $\forall U.D$ | $(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y.\, \langle x, y\rangle \in U^{\mathcal{I}} \rightarrow y \in D^{\mathcal{D}}\}$ |
| datatype atleast | $\geq nU$ | $(\geq nU)^{\mathcal{I}} = \{x \mid \#(\{y.\, \langle x, y\rangle \in U^{\mathcal{I}}\}) \geq n\}$ |
| datatype atmost | $\leq nU$ | $(\leq nU)^{\mathcal{I}} = \{x \mid \#(\{y.\, \langle x, y\rangle \in U^{\mathcal{I}}\}) \leq n\}$ |
| datatype oneOf | $\{v_1, ...\}$ | $\{v_1, ...\}^{\mathcal{I}} = \{v_1^{\mathcal{I}}, ...\}$ |
| **Axiom Name** | **Syntax** | **Semantics** |
| concept inclusion | $C_1 \sqsubseteq C_2$ | $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ |
| object role inclusion | $R_1 \sqsubseteq R_2$ | $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$ |
| object role transitivity | $Trans(R)$ | $R^{\mathcal{I}} = (R^{\mathcal{I}})^{+}$ |
| datatype role inclusion | $U_1 \sqsubseteq U_2$ | $U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$ |
| individual inclusion[7] | $a : C$ | $a^{\mathcal{I}} \in C^{\mathcal{I}}$ |
| individual equality | $a = b$ | $a^{\mathcal{I}} = b^{\mathcal{I}}$ |
| individual inequality | $a \neq b$ | $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ |
| concept existence | $\exists C$ | $\#(C^{\mathcal{I}}) \geq 1$ |

Like description logic, $\mathcal{SHOIN}(D)$ uses concept descriptions to build statements in a DL knowledge base $\mathcal{K}$ (the analogue of an ontology in OWL-DL), which typically comes in two parts: *terminological (TBox)*, and *assertional (ABox)*. In the TBox, we can describe the relevant notions of an application domain by stating properties of concepts and roles and relationships between them. For instance, the notions of agents and arguments are defined in our argumentation framework with the concepts of *Agent* and *Argument* of the ArgCBROnto and the following axioms:

$SocialEntity \sqsubseteq Thing$

$Agent \sqsubseteq SocialEntity$

$Argument \sqsubseteq Thing$

The *properties* of an argument are defined with the roles $hasConclusion$, $promotesValue$, and $hasSupportSet$ and the following axioms and value restrictions:

$Argument \sqsubseteq \forall hasConclusion.Conclusion$

$Argument \sqsubseteq \forall promotesValue.Value$

$Argument \sqsubseteq \forall hasSupportSet.SupportSet$

which state that arguments can have three properties that relate them to objects of the class $Conclusion$, $Value$, and $SupportSet$. Correspondingly, the ABox represents the concrete data of the database $\mathcal{K}$, with the individuals of concepts (instances) and their properties. For instance, the ABox of the ArgCBROnto ontology can include an argument *arg* that promotes a value *solidarity*:

$Argument(arg)$

$promotesValue(arg, solidarity)$

On the other hand, the syntax of the external layer of utterances (locutions) is the same syntax as proposed in (McBurney and Parsons, 2004):

$locution(a_s, \phi)$ or $locution(a_s, a_r, \phi)$

where $Agent(a_s)$ (the sender) and $Agent(a_r)$ (the receiver) are individuals of the $Agent$ concept and $\phi$ is the content of the utterance. The first locution is addressed to all participants in the dialogue, whereas the second is specifically sent to $Agent(a_r)$. We denote the set of well-formed formulae in $\mathcal{SHOIN}(D)$ as $\mathcal{D}$. Then, $\phi \in \mathcal{D}$ can represent statements about problems to solve, evidence about the world, or different types of arguments. Also, we denote the set of individuals members of the concept $Argument$ as $\mathcal{A}$ such that $\forall arg \in \mathcal{A}, Argument(arg)$. Therefore, $\Phi$ is said to be an argument in support of $\phi$ if $\Phi \in \mathcal{A}/\Phi \vdash^{+} \phi$. Correspondingly, $\Phi$ is said to be an argument against $\phi$ if $\Phi \in \mathcal{A}/\Phi \vdash^{-} \phi$.

Also, agents make *propositional commitments* (also known as dialogical commitments) with each locution that they put forward. Therefore, if an agent asserts a locution and another agent challenges it, the first agent has the commitment to provide reasons (or arguments) to justify the validity of that assertion or else, it has to retract it. All commitments made by an agent during the dialogue are commonly stored in an individual database called *commitment store (CS)* (Hamblin, 1970) (there is one commitment store per agent), which is accessible by other agents that are engaged in a dialogue with the agent.

As pointed out above, we follow the standard notation of *modal logics* of knowledge and belief described in (Shoham and Leyton-Brown, 2009, paper 13). Thus, we use the modal operators

$K_i\phi$: "Agent $a_i$ knows $\phi$"

$B_i\phi$: "Agent $a_i$ believes that $\phi$ is true"

$C_g\phi$: "$\phi$ is common knowledge for any agent in the group $g$ if any agent of the group knows it and knows that it is common knowledge"

and the modal connective

$\Diamond\phi$ is satisfied now if $\phi$ is satisfied either now or at some future moment.

Note that here we make a distinction between what agents *know* (which is considered to be true) and what agents *believe* (which forms part of the mental state of an agent and may be true or not). For instance, all farmers that belong to the river basin society of our example *know* that the basin administrator *believes* that avoiding the intervention of a jury will save costs in the water-right transfer process. The farmers know what the administrator believes. However, this doesn't necessarily mean that the basin administrator's opinion is appropriate and, in fact, any farmer can believe that promoting other values may be more appropriate. Therefore, the opinion of the

administrator is subjective and depends on its knowledge; however due to the administrator's power dependency relation over farmers, the farmers have to accept the administrator's point of view.

In addition, as proposed in (McBurney and Parsons, 2004), we use the following simplified elements of *FIPA*'s communicative act library specification[8]:

*Done[locution*$(a_s, \phi)$*, preconditions]*

which indicates that *locution*$(a_s, \phi)$ (or correspondingly *locution*$(a_s, a_r, \phi)$) has been put forward by agent $a_s$ (addressed to agent(s) $a_r$) with content $\phi$ and that the specified *preconditions* hold before this utterance and

*Feasible[condition, locution*$(a_s, \phi)$*]*

which means that if *condition* can take place, *locution*$(a_s, \phi)$ (or correspondingly *locution*$(a_s, a_r, \phi)$) will be put forward by agent $a_s$ (addressed to agent(s) $a_r$) with content $\phi$.

Further notation that we use throughout this paper includes the following:

$a_s$: the $Agent(a_s)$ sender of the locution.

$a_r$: the $Agent(a_r)$ receiver of the locution.

$arg_i$: an $Argument(arg_i)$ of an $Agent(a_i)$.

$SS_i$: the $SupportSet(SS_i)$ of the $Argument(arg_i)$ that has put forward an $Agent(a_i)$.

$CS_i$: the commitment store of an $Agent(a_i)$.

$q$: the $Problem(q)$ under discussion.

$p_i$: the $Solution(p_i)$ (or position) proposed by an $Agent(a_i)$ to solve the $Problem(q)$.

### *4.2. Protocol Syntax*

In this section, we provide the syntax of the communication protocol that the agents of our argumentation framework follow. Therefore, we present the elements of the dialogue: the set of allowed *locutions*, the *commencement rules*, the *combination rules* that govern the course of the dialogue, the *commitment rules* that define the commitments that each agent makes when it utters each locution and how these commitments can be combined, the *rules for speaker order*, and the *termination rules*. The dialogue game presented in this section is aimed at providing a communication protocol for agents that engage in an agreement process. This process can be viewed from several perspectives: as a *collaborative deliberation*, where all agents select the best solution for a problem at hand and do not perceive any reinforcement or reward if their position is selected as the final solution to be applied; as a *negotiation*, where agents try to convince other agents to apply their solution as the best one for solving the problem (with individual *utility functions* that increase their perceived utility); or as a *persuasion*, where each agent tries to persuade the rest of the agents to change their opinions and support its solution as the best option to solve the problem.

*Locutions*

The set of allowed locutions of our dialogue game are the following:

- **L1:** *open_dialogue*$(a_s, \phi)$, where $\phi$ is a problem $q$ to solve in the system application domain. With this locution, an agent $a_s$ opens the argumentation dialogue, asking other agents to collaborate or negotiate to solve a problem that the agent has been presented with.
- **L2:** *enter_dialogue*$(a_s, \phi)$, where $\phi$ is a problem $q$ to solve in the system application domain. With this locution, an agent $a_s$ engages in the argumentation dialogue to solve the problem.
- **L3:** *withdraw_dialogue*$(a_s, \phi)$, where $\phi$ is a problem $q$ to solve in the system application domain. With this locution, an agent $a_s$ leaves the argumentation dialogue to solve the problem.

[8] http://www.fipa.org/specs/fipa00037/SC00037J.html

- **L4:** $propose(a_s, \phi)$, where $\phi$ is a position $p$. With this locution, an agent $a_s$ puts forward the position $p$ as its proposed solution to solve the problem under discussion in the argumentation dialogue.

- **L5:** $why(a_s, a_r, \phi)$, where $\phi$ can be a position $p$ or an argument $arg \in \mathcal{A}$. With this locution, an agent $a_s$ challenges the position $p$ or the argument $arg$ of an agent $a_r$, asking it for a supporting argument.

- **L6:** $noCommit(a_s, \phi)$, where $\phi$ is a position $p$. With this locution, an agent $a_s$ withdraws its position $p$ as a solution for the problem under discussion in the argumentation dialogue.

- **L7:** $assert(a_s, a_r, \phi)$, where $\phi$ can be an argument $arg \in \mathcal{A}$ that supports a position, another argument, or an objectively verifiable evidence about the system application domain. With this locution, an agent $a_s$ sends to an agent $a_r$ an argument or an evidence that supports its position or a previous argument that $a_r$ has put forward.

- **L8:** $accept(a_s, a_r, \phi)$, where $\phi$ can be an argument $arg \in \mathcal{A}$ or a position $p$ to solve a problem. With this locution, an agent $a_s$ accepts the argument $arg$ or the position $p$ of an agent $a_r$. Also, this locution can be used at the end of the dialogue to inform all agents about the final position agreed upon as the best position to solve the problem. In that case, $a_r$ denotes *all* individuals that belong to the concept *Agent*, except for the sender $a_s$ ($all: \forall a_i, a_i \neq a_s / Agent(a_i)$).

- **L9:** $attack(a_s, a_r, \phi)$, where $\phi$ is an argument $arg \in \mathcal{A}$ of an agent $a_s$. With this locution, an agent $a_s$ challenges an argument of an agent $a_r$ with its argument $arg$.

- **L10:** $retract(a_s, a_r, \phi)$, where $\phi$ is an argument $arg \in \mathcal{A}$. With this locution, an agent $a_s$ informs an agent $a_r$ that it withdraws the argument $arg$ that it put forward in a previous step of the argumentation dialogue.

*Commencement Rules*

The dialogue starts when an agent $a_s$ is presented with a new problem $q$ to solve. First, the agent tries to solve it by using its own knowledge resources. Then, it opens a dialogue with other agents by sending them the locution $open\_dialogue(a_s, a_r, q)$, where $a_r$ can be any agent $a_i$ that $a_s$ knows. After that, $a_i$ enters in the dialogue by posing the locution $enter\_dialogue(a_s, q)$ (where $a_s = a_i$). After that, if $a_i$ has been able to find a solution for $q$, it proposes this initial position $p$ to solve the problem $q$ with the locution $propose(a_s, p)$ (where $a_s = a_i$) and waits for the challenges of other agents or for other position proposals. Otherwise, $a_i$ can challenge the positions of other agents engaged in the dialogue with the locution $why(a_s, a_r, p)$ (where $a_s = a_i$).

*Rules for the Combination of Locutions*

The rules for the combination of locutions define which locution can be put forward at each step of the dialogue game. Figure 3 represents a state machine with the possible stages of our dialogue game protocol. As shown in the figure, the protocol has three main stages: the *opening* stage, where the agent that initiates the dialogue opens the argumentation process to solve a problem; the *argumentation* stage, where agents argue to reach an agreement about the best solution to apply to solve the problem; and the *closing* stage, where the final decision about the position selected to solve the problem is reported to all agents that have participated in the dialogue. The stages of our dialogue game and the rules for the combination of locutions in each stage are presented below.

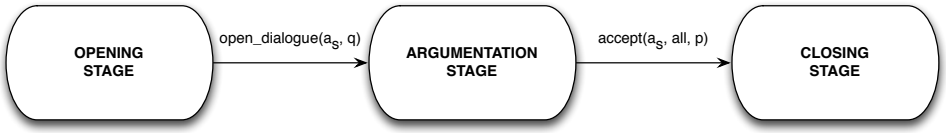


Figure 3: State Machine of the Dialogue Game

**Opening Stage:**

The opening stage commences when an agent $a_s$ wants to establish an agreement process with other agents to solve a problem $q$ that it has been faced with. Then, it uses the locution $open\_dialogue(a_s, q)$ to start the dialogue.

**Argumentation Stage:**

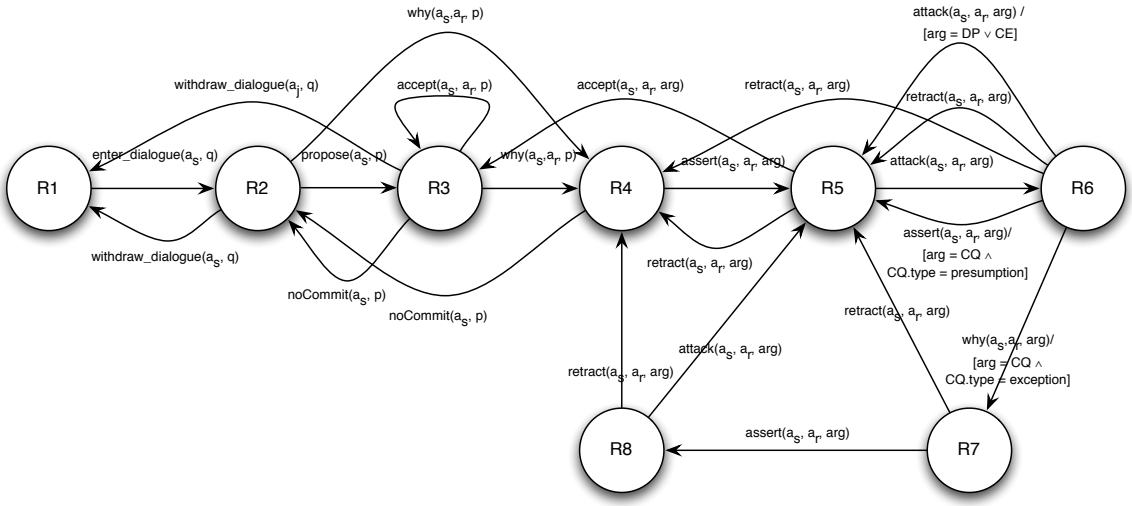


Figure 4: State Machine of the Argumentation Stage

The argumentation stage follows the opening stage. Here, agents argue to reach an agreement about the solution to apply to the problem $q$. As shown in Figure 4, this stage is divided into a set of substages whose activation is defined by the following rules (for reasons of clarity, substages are labelled with the name of the rule that applies in each case):

- **R1:** Once the dialogue has been opened, any agent that has been informed about it can enter in by using the locution $enter\_dialogue(a_s, q)$.

- **R2:** After entering the dialogue, an agent can propose its position $p$ to solve the problem $q$ by putting forward the locution $propose(a_s, p)$. Alternatively, the agent can challenge the positions of other agents engaged in the dialogue (without its own position being proposed) with the locution $why(a_s, a_r, p)$. Also, in this substage, the agent can withdraw from the dialogue by using the locution $withdraw\_dialogue(a_s, q)$.

- **R3:** In this substage, an agent that has proposed its position $p$ to solve the problem $q$ can be asked by another agent for an argument to support this position with the locution $why(a_s, a_r, p)$. Also, $p$ can be accepted by an agent engaged in the dialogue, who reports to the proponent agent with the locution $accept(a_s, a_r, p)$. Furthermore, the proponent agent can withdraw its position $p$ with the locution $noCommit(a_s, p)$. Alternatively, it can leave the dialogue with the locution $withdraw\_dialogue(a_s, q)$.

- **R4:** After being asked for an argument to support its position $p$, an agent can use its knowledge resources to provide the requester agent with this argument $arg$ by means of the locution $assert(a_s, a_r, arg)$. Alternatively, it can withdraw its position $p$ by using the locution $noCommit(a_s, p)$.

- **R5:** An agent that has received a support or an attack argument from another agent can use its knowledge resources to create an attack argument $arg$ and send it to the other agent with the locution $attack(a_s, a_r, arg)$. Also, the agent can accept the supporting argument and report to the other agent with the locution $accept(a_s, a_r, arg)$, where $arg$ is the supporting argument received. In its turn, an agent that has asserted the argument $arg$ can withdraw it with the locution $retract(a_s, a_r, arg)$.

- **R6:** When an agent receives an attack argument from another agent, it analyses the type of the attack and can use its knowledge resources to try to rebut the attack. Therefore, if the attacking argument $arg$ was a distinguishing premise or a counter-example ($arg = (DP \vee CE)$), the agent can distinguish the argument of the other agent with other distinguishing premise or else counter-attack with another counter-example by using the locution $attack(a_s, a_r, arg)$. If the attacking argument was a critical question of the type presumption ($arg = CQ \wedge CQ.type = presumption$), the agent can use its knowledge resources to create and show the other agent an argument $arg$ with evidence that supports that presumption by using the locution $assert(a_s, a_r, arg)$. Finally, if the attacking argument was a critical question of the type exception ($arg = CQ \wedge CQ.type =$

*exception*), the agent can ask the other agent for an argument to support this critical question by stating the locution $why(a_s, a_r, arg)$. Alternatively, if the agent cannot rebut the attack, it can retract its argument with the locution $retract(a_s, a_r, arg)$. In its turn, any agent that has asserted the argument $arg$ can withdraw it with the locution $retract(a_s, a_r, arg)$.

- **R7:** If an agent is asked by another agent to provide a supporting argument for its critical question of the type exception, this agent must use the locution $assert(a_s, a_r, arg)$ to assert an argument $arg$ with evidence to support this critical question attack or else retract the attack by putting forward the locution $retract(a_s, a_r, arg)$.

- **R8:** Once an agent has been provided by another agent with evidence that supports the other agent's critical question of the type exception, the fist agent can retract its argument $arg$ and report to the other agent with the locution $retract(a_s, a_r, arg)$ or else can try to generate an attack argument $arg$ for the other agent's argument and send it the locution $attack(a_s, a_r, arg)$.

Also, note that any agent can withdraw its position at any stage of the dialogue. It implies that there is a transaction labelled with the locution $noCommit(a_s, p)$ from substages $R5...R8$ to substage $R2$. However, these substages do not appear in Figure 4 for reasons of clarity.

**Closing Stage:**

The closing stage can be activated at any time in the dialogue by the agent $a_i$ that opened it. This stage is reached by putting forward the locution $accept(a_s, all, p)$ (where $a_s = a_i$), which informs all the participating agents about the final position $p$ agreed upon as the solution for the problem $q$. Here, the commitment store of all agents is deleted.

*Commitment Rules*

As pointed out above, agents make *dialogical commitments* with each locution that they put forward. These commitments are stored in an individual commitment database called *commitment store (CS)*. Also, the inclusion of a new commitment in the commitment store can make previous commitments be inconsistent or invalid. The commitment rules that define the commitments associated with each locution and how their inclusion in the commitment store affects previous commitments are presented below.

- **CR1:** The locution $enter\_dialogue(a_s, q)$ gives rise to the creation of the commitment store $CS_s$ of the sender agent.

- **CR2:** The locution $propose(a_s, p)$ inserts the position $p$ into the commitment store $CS_s$ of the sender agent. If there is a previous position in $CS_s$, this position is replaced with the new position $p$. Thus, only one position can prevail in any commitment store.

- **CR3:** The locution $withdraw\_dialogue(a_s, q)$ deletes the commitment store $CS_s$ of the sender agent. This implies that the final agreement is only taken among the agents that remain listening in the substages $R2$ or $R3$. Also, agents cannot withdraw the dialogue before withdrawing any position that they have proposed with the locution $noCommit(a_s, p)$.

- **CR4:** The locution $accept(a_s, a_r, p)$ inserts the position $p$ into the commitment store $CS_s$ of the sender. If there is a previous position in $CS_s$, this position is replaced with the new position $p$.

- **CR5:** The locution $noCommit(a_s, p)$ deletes $p$ from the commitment store $CS_s$ of the sender.

- **CR6:** The locution $why(a_s, a_r, p)$ commits the receiver to provide the sender with a supporting argument $arg$ for $p$ or else to withdraw $p$ with the locution $noCommit(a_s, p)$.

- **CR7:** The locution $assert(a_s, a_r, arg)$ inserts the argument $arg$ in the commitment store $CS_s$ of the sender. Also, commitment stores cannot have inconsistent arguments. Therefore, if the conclusion of $arg$ contradicts the conclusion of a previous argument stored in $CS_s$, the sender cannot put forward the locution $assert(a_s, a_r, arg)$ before deleting the inconsistent argument from $CS_s$ with the locution $retract(a_s, a_r, arg)$ addressed to any agent that is maintaining a dialogue with the sender. Furthermore, if $arg$ includes in its support set an argumentation scheme with a critical question of the type presumption, the locution $assert(a_s, a_r, arg)$ commits the sender to provide evidence to support this argument if another agent attacks it with the locution $attack(a_s, a_r, arg)$, where $arg$ includes such critical question, or else to retract the argument.

- **CR8:** The locution $accept(a_s, a_r, arg)$ inserts the argument $arg$ into the commitment store $CS_s$ of the sender. Again, commitment stores cannot have inconsistent arguments. Therefore, if the conclusion of $arg$ contradicts the conclusion of a previous argument stored in $CS_s$, the sender cannot put forward the locution *assert($a_s$, $a_r$, arg)* before deleting the inconsistent argument from $CS_s$ with the locution *retract($a_s$, $a_r$, arg)* addressed to any agent that is maintaining a dialogue with the sender.

- **CR9:** The locution $retract(a_j, a_k, arg)$ deletes the argument $arg$ from the commitment store $CS_j$ of $a_j$.

- **CR10:** The locution $attack(a_s, a_r, arg)$ inserts the argument $arg$ in the commitment store $CS_s$ of the sender. As pointed out above, commitment stores cannot have inconsistent arguments. Therefore, if the conclusion of $arg$ contradicts the conclusion of a previous argument stored in $CS_s$, the sender cannot put forward the locution $attack(a_s, a_r, arg)$ before deleting the inconsistent argument from $CS_s$ with the locution $retract(a_s, a_r, arg)$ addressed to any agent that is maintaining a dialogue with the sender. Also, if $arg$ includes an argumentation scheme with a critical question of the type exception, the locution $attack(a_s, a_r, arg)$ commits the sender to provide an evidence to support this attack if another agent challenges this exception with the locution $why(a_s, a_r, arg)$, or else to retract it.

- **CR11:** The locution $accept(a_s, all, p)$ $(all : \forall a_i, a_i \neq a_s \mid Agent(a_i))$ deletes the commitment stores of all agents that are still participating in the dialogue (including the initiator). This is a special case of commitment rule that grants the initiator to manage the commitment stores of other agents and ensures an ordered termination of the dialogue. Thus, we assume the existence of a normative level that all participants agree upon before they are able to enter in the dialogue.

*Rules for Speaker Order*

During the dialogue, agents take turns putting forward locutions. Each time an agent $a_s$ sends a locution to another agent $a_r$, it waits for an answer from $a_r$. However, any agent can hold parallel argumentation dialogues with several agents. Thus, in each of these dialogues, the argumentation succeeds as a two-party dialogue between two agents, one agent sending a locution to the other agent and waiting for a response. Nevertheless, the locution $open\_dialogue(a_s, q)$ is received by all agents of the society $S_t$. The locutions $accept(a_s, all, p)$, $propose(a_s, p)$, $noCommit(a_s, p)$ and $withdraw\_dialogue(a_s, p)$ are received by all of the agents that are engaged in the dialogue. With these locutions, the sender agent does not expect any response.

In this dialogue game protocol, we assume that all participating agents can always see the positions of the other agents by looking at their commitment stores. Also, when two agents are engaged in a dialogue, each agent has full view to the commitment store of the other agent. In this way, these agents can see the commitments associated to the arguments of their partners, but other agents can only see to the positions proposed by each agent in the dialogue (which are also stored in the commitment stores). This preserves the privacy of the arguments that an agent puts forward in its argumentation dialogue with another agent. Note that if an agent wants to ask other agents for an opinion about an argument that it has received, it simply has to send those agents the argument, as if the argument was its own. This simple rule allows us to use the same dialogue game to govern collaborative deliberations, persuasion dialogues, and negotiations. In the collaborative deliberations, all agents follow the common objective of proposing the best solution for a problem at hand. Therefore, there are no agents interested in trying to take advantage of the information interchanged between other agents to obtain a greater benefit with the final agreement reached. However, this could be the case in a persuasion or a negotiation, where each agent tries to persuade other agents to change their point of view or tries to increase its perceived utility value with the final agreement, thereby using any extra information about other agents' knowledge and preferences in order to achieve that.

*Termination Rules*

The normal termination of the dialogue occurs when the argumentation process ends with all participating agents having proposed a prevailing position or having accepted the position of another agent. Then, agents may reach a decision about the final solution for the problem under discussion. In the ideal case, only the position of one participating agent prevails, while the other agents have withdrawn theirs and accepted this position by using the locution $accept(a_s, a_r, p)$. However, if at the end of the dialogue more than one position is still undefeated, agents can use a voting mechanism (selecting the position most accepted) or a random selection to decide the final outcome of the agreement process.

In any case, the agent $a_i$ that opened the dialogue is responsible for reporting to all participating agents the final position $p$ that has been selected as solution for the problem $q$ at hand, by using the locution $accept(a_s, all, p)$

(where $a_s = a_i$). To avoid infinite dialogues, agents cannot put forward the same argument twice during a dialogue with another agent, unless new pieces of evidence are available. Furthermore, a maximum time to reach an agreement can be established and agents must accept a position among those available at that moment to solve the problem.

Note that agents can maintain several parallel dialogues with other agents. Thus, once an agent has entered in the argumentation process with the locution *enter_dialogue*($a_s$, $q$), it remains waiting to propose a position in substage $R2$ or listening to incoming locutions of other agents in substage $R3$. Then, the specific dialogue with an agent that has asked another agent for a supporting argument for its position $p$ continues the subsequent substages, but the agent still remains in $R3$ listening to other requests. Finally, the locution *noCommit*($a_s$, $p$) commits the sender to terminate any dialogue that it has started to defend $p$.

### *4.3. Semantics*

In this section, we provide the formal semantics for the locutions of our dialogue game protocol. This semantics provides a common understanding about the properties of the communication language between agents. There are different methods for providing a communication language with a semantics (Tennent, 1991), for instance, the *operational* approach followed in this paper.

Operational semantics views the dialogue game protocol as an abstract state machine and precisely defines the transitions between states. These transitions are triggered by the utterance of each locution. However, from some stages, an agent can utter different locutions following different *agent decision mechanisms*, which are reasoning mechanisms that agents can use to choose the locution to utter in the next step of the dialogue among a set of candidates. These mechanisms depend on the knowledge that agents can infer from their knowledge resources or even on the specific design of agents. For instance, agents that are designed to be more competitive and, if possible, always put forward attack arguments or agents that are designed to remain listening and only engage in a dialogue if their positions or arguments are attacked. Figure 5 shows the decision mechanisms that agents can use in each substage of the argumentation stage of our protocol. For purposes of clarity , the arrows labelled with the decision mechanism *D8* (presented below) from substages *R5*, *R6*, *R7*, and *R8* to substage *R2* are omitted in the figure.

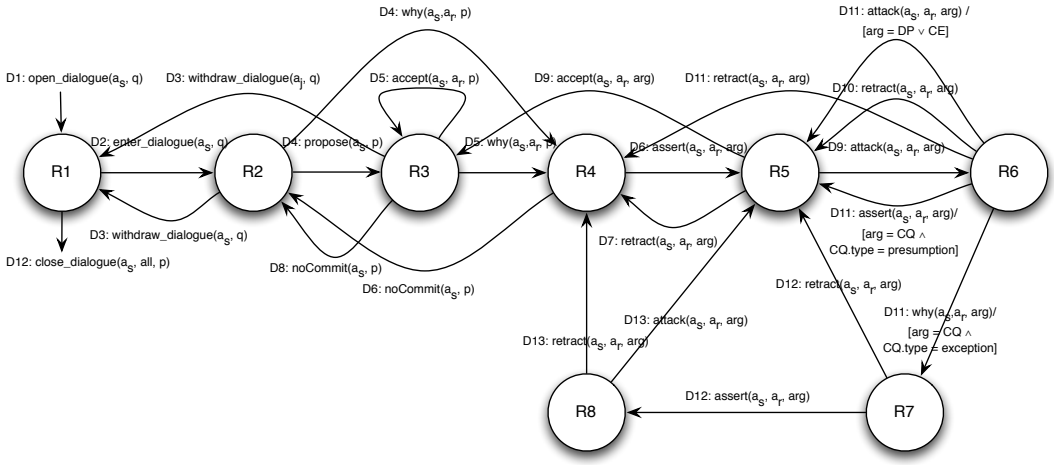


Figure 5: Decision Mechanisms of the Dialogue Game

To define the transition rules of our protocol we follow the notation of (McBurney and Parsons, 2004):

$$\langle a_i, K, o \rangle$$

where $a_i$ is an agent, $K$ is a decision mechanism (or the terminal state $T$), and $o$ is the output of the mechanism $K$ (send a locution or remain listening to incoming locutions). Some transitions are labelled with the locutions that trigger them while others (which occur between the mechanisms of a single agent) remain unlabeled. Also, if no specific output is invoked, we denote this by a period in the third parameter of the triple ($\langle a_i, K, . \rangle$).

Specifically, we have identified the following decision mechanisms:

- **D1 Open Dialogue:** A mechanism that allows an agent to open a dialogue with other agents of the society $S_t$ that the agent belongs to, by uttering or not uttering the locution $open\_dialogue(a_s, q)$. The output of this mechanism is: *send(open_dialogue($a_s$, $\phi$))*.

- **D2 Enter or Close Dialogue:** A mechanism that allows an agent to decide to engage in a dialogue and utter or not utter the locution $enter\_dialogue(a_s, q)$. By this mechanism, the agent makes a query to its knowledge resources, trying to find a solution for the problem to solve. If the agent can provide a solution for the problem, the agent uses the mechanism to decide whether or not it enters in the dialogue. Alternatively, the agent that started the dialogue can also close it with the locution $accept(a_s, all, p)$. The outputs of this mechanism are: *send(enter_dialogue($a_s$, $\phi$))*, *listen()*, or *send(close_dialogue($a_s$, all, $\phi$))*.

- **D3 Withdraw from Dialogue:** A mechanism that allows an agent to withdraw from the dialogue and put forward the locution $withdraw\_dialogue(a_s, q)$. The mechanism first checks that the agent does not have any active position to solve the problem (agents cannot withdraw from the dialogue before withdrawing their positions). Possible outputs are: *send(withdraw_dialogue($a_s$, $\phi$))*.

- **D4 Propose or Challenge:** A mechanism that allows an agent to make a proposal to solve the problem under discussion and utter the locution $propose(a_s, p)$ or to challenge the positions of other agents by uttering the locution $why(a_s, a_r, p)$. By this mechanism the agent uses its knowledge resources to generate and select the position to propose. If the agent has been able to generate a position to solve the problem, it uses the mechanism to decide whether to put forward that position. In any case, the agent can challenge other positions or remain listening to the utterances of other agents. The outcomes for this mechanism are: *send(propose($a_s$, $\phi$))*, *send(why($a_s$, $a_r$, $\phi$))*, or *listen()*.

- **D5 Accept or Challenge:** A mechanism that allows an agent to query its knowledge resources and decide to accept or challenge the position of another agent. If the agent is able to generate the same position as its candidate to solve the problem, it can utter the locution $accept(a_s, a_r, p)$ to accept the other's position. Else, if the position cannot be generated or is generated but not ranked as the most suitable solution for the problem, the agent can use this mechanism and decide to accept the other agent's position or to challenge it with the locution $why(a_s, a_r, p)$. Thus, possible outcomes are: *send(accept($a_s$, $\phi$))* or *send(why($a_s$, $a_r$, $\phi$))*.

- **D6 Defend Position:** A mechanism that allows an agent to defend its position from a challenge or else, to withdraw it. By this mechanism the agent decides if it is able to use its knowledge resources to provide the challenger with an argument that supports its position. In that case, it can utter the locution $assert(a_s, a_r, arg)$. Otherwise, the agent has to withdraw the position by using the locution $noCommit(a_s, p)$. Also, the agent that put forward the challenge can use this mechanism to listen for the answer to its challenge. The outcomes of this mechanism are: *send(assert($a_s$, $a_r$, $\phi$))*, *send(noCommit($a_s$, $\phi$))* or *listen()*.

- **D7 Withdraw Argument:** This mechanism allows an agent to decide whether to withdraw an argument that it has put forward, using the locution $retract(a_s, a_r, \phi)$. Possible outcomes are: *send(retract($a_s$, $a_r$, $\phi$))*.

- **D8 Withdraw Position:** A mechanism that allows an agent to decide whether to withdraw its proposed position with the locution $noCommit(a_s, p)$. The output of this mechanism is: *send(noCommit($a_s$, $\phi$))*.

- **D9 Accept or Attack:** A mechanism that allows an agent to query its knowledge resources and decide to accept or attack the argument of other agent. If the argument is consistent with the information inferred from the knowledge resources of the agent, it can utter the locution $accept(a_s, a_r, arg)$ to accept the other's argument. Otherwise, if the argument is inconsistent and an attack argument can be generated from the knowledge resources, the agent can use this mechanism to decide to attack the argument by uttering the locution $attack(a_s, a_r, arg)$. Otherwise, if the argument cannot be decided (there is not enough information in the knowledge resources to support or rebut the argument), the agent also accepts it. Thus, possible outcomes are: *send(accept($a_s$, $\phi$))* or *send(attack($a_s$, $a_r$, $\phi$))*.

- **D10 Withdraw Attack:** This mechanism allows an agent to decide whether to withdraw an attack that it has put forward, using the locution $retract(a_s, a_r, \phi)$. Possible outcomes are: *send(retract($a_s$, $a_r$, $\phi$))* or *listen()*.

- **D11 Rebut Attack:** A mechanism that allows an agent to rebut an attack to its argument. By this mechanism, the agent evaluates the attack argument received and queries its knowledge resources to search for

information that supports or rebuts the attack. If the attack argument poses a critical question of the type *presumption*, the agent can rebut the attack by showing information that supports its argument with the locution $assert(a_s, a_r, \phi)$. If the attack argument poses a critical question of the type *exception*, the agent can rebut the attack by challenging it with the locution $why(a_s, a_r, \phi)$. Otherwise, if the attack argument poses a distinguishing-premise or a counter-example to the agent's argument, it can use the locution $attack(a_s, a_r, arg)$ to rebut the attack by counter-attacking with another distinguishing-premise or counter-example. In any case, if the agent is not able to rebut the attack with the information inferred from its knowledge resources, it can retract its argument by uttering the locution $retract(a_s, a_r, \phi)$. Therefore, the outcomes of this mechanism are: *send(assert($a_s$, $a_r$, $\phi$))*, *send(why($a_s$, $a_r$, $\phi$))*, *send(attack($a_s$, $a_r$, $\phi$))*, or *send(retract($a_s$, $a_r$, $\phi$))*.

- **D12 Defend Argument:** This mechanism allows an agent to rebut a challenge to its argument, which poses a critical question of the type *exception*. With this mechanism, the agent queries its knowledge resources and tries to find information that supports its attack argument. In that case, the agent can rebut the attack by showing this information uttering the locution $assert(a_s, a_r, arg)$. Otherwise, the agent has to withdraw the attack by uttering $retract(a_s, a_r, arg)$. Also, the agent that put forward the challenge can use this mechanism to listen for the answer to its challenge. Possible outcomes are: *send(assert($a_s$, $a_r$, $\phi$))*, *send(retract($a_s$, $a_r$, $\phi$))*, or *listen()*.

- **D13 Retract or Attack:** This mechanism allows an agent to counter-attack a critical question attack of the type *exception* posed to its argument. With this mechanism, the agent queries its knowledge resources to search for information that rebuts the attack. Then, if the agent finds this information, it can counter-attack by uttering the locution $attack(a_s, a_r, \phi)$. Otherwise, the agent has to withdraw its argument by uttering the locution $retract(a_s, a_r, \phi)$. Thus, the outcomes of the mechanism are: *send(attack($a_s$, $a_r$, $\phi$))* or *send(retract($a_s$, $a_r$, $\phi$))*.

Table 4 shows the transition rules of the operational semantics of our protocol.

| | |
|---|---|
| **TR1:** | $\langle a_s, D1, send(open_dialogue(a_s, \phi))\rangle \xrightarrow{L1} \langle a_s, D2, .\rangle$ |
| **TR2:** | $\langle a_s, D2, send(enter_dialogue(a_s, \phi))\rangle \xrightarrow{L2} \langle a_s, D3, .\rangle$ |
| **TR3:** | $\langle a_s, D2, send(enter_dialogue(a_s, \phi))\rangle \xrightarrow{L2} \langle a_s, D4, .\rangle$ |
| **TR4:** | $\langle a_s, D2, listen()\rangle \rightarrow \langle a_s, D2, .\rangle$ |
| **TR5:** | $\langle a_s, D2, send(close_dialogue(a_s, all, \phi))\rangle \xrightarrow{L8} \langle all, T, .\rangle$ |
| **TR6:** | $\langle a_s, D3, send(withdraw_dialogue(a_s, \phi))\rangle \xrightarrow{L3} \langle a_s, D2, listen()\rangle$ |
| **TR7:** | $\langle a_s, D4, send(propose(a_s, p))\rangle \xrightarrow{L4} \langle a_s, D8, .\rangle$ |
| **TR8:** | $\langle a_s, D4, send(propose(a_s, p))\rangle \xrightarrow{L4} \langle a_s, D5, .\rangle$ |
| **TR9:** | $\langle a_s, D4, send(propose(a_s, p))\rangle \xrightarrow{L4} \langle a_r, D5, .\rangle$ |
| **TR10:** | $\langle a_s, D4, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_s, D4, listen()\rangle$ |
| **TR11:** | $\langle a_s, D4, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_r, D6, .\rangle$ |
| **TR12:** | $\langle a_s, D4, listen()\rangle \rightarrow \langle a_s, D4, .\rangle$ |
| **TR13:** | $\langle a_s, D8, send(noCommit(a_s, \phi))\rangle \xrightarrow{L6} \langle a_s, D4, listen()\rangle$ |
| **TR14:** | $\langle a_s, D8, send(noCommit(a_s, \phi))\rangle \xrightarrow{L6} \langle a_s, D3, .\rangle$ |
| **TR15:** | $\langle a_s, D5, send(accept(a_s, a_r, \phi))\rangle \xrightarrow{L8} \langle a_s, D5, .\rangle$ |
| **TR16:** | $\langle a_s, D5, send(accept(a_s, a_r, \phi))\rangle \xrightarrow{L8} \langle a_r, D5, .\rangle$ |
| **TR17:** | $\langle a_s, D5, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_s, D6, listen()\rangle$ |
| **TR18:** | $\langle a_s, D5, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_r, D6, .\rangle$ |
| **TR19:** | $\langle a_s, D6, listen()\rangle \rightarrow \langle a_s, D6, .\rangle$ |
| **TR20:** | $\langle a_s, D6, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D7, .\rangle$ |
| **TR21:** | $\langle a_s, D6, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D8, .\rangle$ |

Continues on the next page

| | |
|---|---|
| **TR22:** | $\langle a_s, D6, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_r, D9, .\rangle$ |
| **TR23:** | $\langle a_s, D6, send(noCommit(a_s, \phi))\rangle \xrightarrow{L6} \langle a_s, D3, .\rangle$ |
| **TR24:** | $\langle a_s, D6, send(noCommit(a_s, \phi))\rangle \xrightarrow{L6} \langle a_s, D4, listen()\rangle$ |
| **TR25:** | $\langle a_s, D7, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D6, .\rangle$ |
| **TR26:** | $\langle a_s, D9, send(accept(a_s, a_r, \phi))\rangle \xrightarrow{L8} \langle a_s, D3, .\rangle$ |
| **TR27:** | $\langle a_s, D9, send(accept(a_s, a_r, \phi))\rangle \xrightarrow{L8} \langle a_s, D5, .\rangle$ |
| **TR28:** | $\langle a_s, D9, send(accept(a_s, a_r, \phi))\rangle \xrightarrow{L8} \langle a_r, D8, .\rangle$ |
| **TR29:** | $\langle a_s, D9, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_s, D10, .\rangle$ |
| **TR30:** | $\langle a_s, D9, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_r, D8, .\rangle$ |
| **TR31:** | $\langle a_s, D9, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_r, D11, .\rangle$ |
| **TR32:** | $\langle a_s, D10, listen()\rangle \rightarrow \langle a_s, D10, .\rangle$ |
| **TR33:** | $\langle a_s, D10, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D9, .\rangle$ |
| **TR34:** | $\langle a_s, D10, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D7, .\rangle$ |
| **TR35:** | $\langle a_s, D10, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D8, .\rangle$ |
| **TR36:** | $\langle a_s, D11, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D7, .\rangle$ |
| **TR37:** | $\langle a_s, D11, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D8, .\rangle$ |
| **TR38:** | $\langle a_s, D11, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_r, D9, .\rangle$ |
| **TR39:** | $\langle a_s, D11, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_s, D12, listen()\rangle$ |
| **TR40:** | $\langle a_s, D11, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_r, D8, .\rangle$ |
| **TR41:** | $\langle a_s, D11, send(why(a_s, a_r, \phi))\rangle \xrightarrow{L5} \langle a_r, D12, .\rangle$ |
| **TR42:** | $\langle a_s, D11, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_s, D7, .\rangle$ |
| **TR43:** | $\langle a_s, D11, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_s, D8, .\rangle$ |
| **TR44:** | $\langle a_s, D11, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_r, D9, .\rangle$ |
| **TR45:** | $\langle a_s, D11, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D6, .\rangle$ |
| **TR46:** | $\langle a_s, D11, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D6, listen()\rangle$ |
| **TR47:** | $\langle a_s, D12, listen()\rangle \rightarrow \langle a_s, D12, .\rangle$ |
| **TR48:** | $\langle a_s, D12, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D8, .\rangle$ |
| **TR49:** | $\langle a_s, D12, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_r, D13, .\rangle$ |
| **TR50:** | $\langle a_s, D12, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D7, .\rangle$ |
| **TR51:** | $\langle a_s, D12, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D8, .\rangle$ |
| **TR52:** | $\langle a_s, D12, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D9, .\rangle$ |
| **TR53:** | $\langle a_s, D13, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_s, D7, .\rangle$ |
| **TR54:** | $\langle a_s, D13, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_s, D8, .\rangle$ |
| **TR55:** | $\langle a_s, D13, send(attack(a_s, a_r, \phi))\rangle \xrightarrow{L9} \langle a_r, D9, .\rangle$ |
| **TR56:** | $\langle a_s, D13, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_s, D6, .\rangle$ |
| **TR57:** | $\langle a_s, D13, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D6, listen()\rangle$ |

Table 4: Transition Rules of the Dialogue Game Protocol.

These transition rules provide the operational semantics of the dialogue, defining the range of potential decisions that agents can make in each stage of the dialogue. In section 5, an example of the water-right transfer scenario is provided to illustrate the performance of the dialogue game protocol proposed in this section.

### 4.4. Protocol Evaluation

There are several ways to evaluate a dialogue game protocol, although there are no standard methods agreed by the argumentation in artificial intelligence research community (McBurney and Parsons, 2009). In (Heras et al., 2013), we run several experiments to evaluate our argumentation framework by simulating an agreement process in a real domain. All experiments were implemented by using our protocol and by using protocols without argumentation. This permitted to identify the circumstances under which the use of our argumentation system (and implicitly, our dialogue game protocol) produces an improvement on the percentage of times that an agreement is reached and the number of agents that is persuaded. In (McBurney et al., 2002) McBurney et al. provided a criteria to assess a dialogue game protocol for agent interactions, proposing a set of desiderata that protocols of this type should satisfy. These desiderata draw on research in agent interaction, on criteria for assessment of automated auction mechanisms, and on elements of argumentation theory and political theory. In this section, we discuss that our dialogue game protocol satisfies the desiderata following this approach. Also, by conforming with these desiderata, our dialogue game protocol produces outcomes that are Pareto optimal, i.e., that any other outcome leaves at least one participant worse off (demonstration available at *McBurney02c*).

- Stated Dialogue Purpose: the purpose of the dialogue is to reach an agreement to provide the best solution for a problem. All participants are aware of this purpose before they enter in the dialogue. The syntax requires the agent that opens the argumentation dialogue to use the locution $open_dialogue$ to inform other agents of the problem to solve and to ask them to collaborate.

- Diversity of individual purposes: all agents entering the dialogue can have a different position about the best solution for the problem at hand. The protocol also permits agents to enter in the dialogue to express their view about other agents' positions, even if they are not able to provide their own solution. Then, the syntax and semantics of the protocol allow agents to defend their individual positions and reach an agreement about the best solution to apply.

- Inclusiveness: agents participating in the agreement process must agree on a set of norms that control the behaviour of the society that agents belong. Assuming that agents observe these norms, the protocol allows any potential agent that is qualified and willing to participate to engage in the dialogue.

- Transparency: protocol syntax and semantics are public and available to all participants, so they know the rules and structure of the dialectical system prior to commencement of the dialogue.

- Fairness: locutions, rules and semantics of the protocol are the same for all participants except for the initiator of the dialogue, which has the extra responsibilities of starting the dialogue process and conveying the information about the final outcome. This is known by the other participants, does not affect its performance as dialogue participant, and does not grant this agent any privileges over their partners.

- Clarity of Argumentation Theory: protocol syntax and semantics conforms to the argumentation theory formed by our case-based argumentation framework, the knowledge resources of our framework, and the argument ordering established by our defeat relation over arguments (Heras, 2011, Definition 3.5.5). The commitment rules of our protocol explicitly establish the commitments associated with each locution and how their inclusion in the commitment store affects previous commitments. The rules for the combination of locutions define which locution can be put forward at each step of the dialogue game, allowing agents to agree on rules of inference and procedure, and have reasonable expectations of the responses of others.

- Separation of Syntax and Semantics: syntax and semantics are defined separatedly and are publicly available to all participants.

- Rule-Consistency: all protocol rules are consistent with the syntax and semantics.

- Encouragement of Resolution: the rules for the combination of locutions guide the dialogue to reach an agreement over a specific position. Termination rules ensure an outcome of the dialogue and avoid infinite loops. However, if the process ends on a disagreement (when agents do no have more positions and arguments to put forward, more than one position is still undefeated), agents can use a voting mechanism (selecting the position most accepted) or a random selection to decide the final outcome of the dialogue. In addition, although a maximum time to reach an agreement can be established, the rules of the protocol ensure that prevailing positions at each time are those in which more agents agree upon.

- Discouragement of Disruption: termination rules preclude disruptive behaviour, such as uttering the same locution to put forward the same argument twice during a dialogue with the same agent (if no new evidences have emerged). Also, the rules for the combination of locutions allow agents to withdraw their positions and arguments, and to leave the dialogue in an orderly manner.

- Enablement of Self-Transformation: the locutions of the protocol and the rules for their combination allow agents to change their positions and arguments during the dialogue. Agents are able to withdraw positions and arguments, retracting from their associated commitments by means of the commitment rules.

- System Simplicity: the protocol is quite simple, including only 10 locutions and 8 rules for their combination. In each stage of the dialogue, only a set of locutions are permitted. Agents take turns to make locutions in two-party dialogues, but each agent can hold parallel argumentation dialogues with several participants.

- Computational Simplicity: the simulation experiments of our argumentation framework presented in (Heras et al., 2013) implicitly show that our dialogue game protocol allow agents to reach agreements with a reasonable amount of locutions interchanged between them (a total average of less than 40 locutions interchanged in a dialogue among 9 agents with more than 30 cases in each agent case-base).

## 5. Water-Right Transfer Example

This section illustrates the dialogue game protocol presented in this paper by means of an example in the water-right transfer domain (as introduced in Section 2). In this example, 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, as shown in Figure 6 the premises of the original problem could 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*).

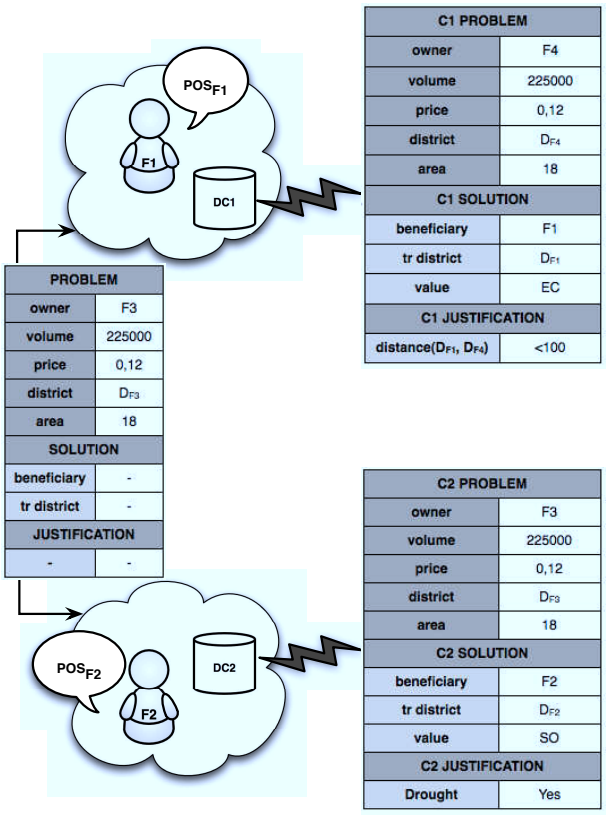


Figure 6: Generation of Positions

After the opening of the trading table by the market facilitator, in the first step of the argumentation process, the basin administrator *BA* opens the dialogue to solve the water-right transfer problem. Thus, it sends the locution *open_dialogue(BA, q)* (where *q* contains the premises of the problem) to all agents of the group, which is the river basin *RB*. Then, the BA enters in the dialogue by putting forward the locution *enter_dialogue(BA, q)*. Figure 7 shows the sequence of locutions interchanged by the agents during the dialogue.

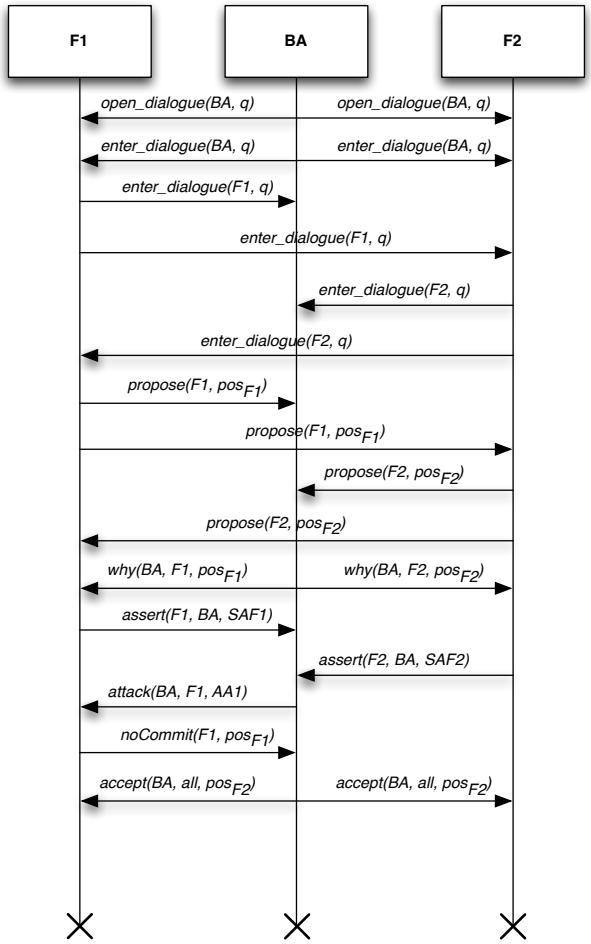


Figure 7: Sequence of Locutions

Assuming that both farmers $F1$ and $F2$ are interested in entering in the dialogue and arguing to win the transfer, they will assert the locutions *enter_dialogue(F1, q)* and *enter_dialogue(F2, q)*, respectively. After that, they will search for domain-cases in their case-bases ($DC1$ and $DC2$, respectively) to generate their potential positions. To query the case-bases, the problem is formatted as a target case without solution and justification, as shown on the left side of Figure 6. In this case, the solution consists of the identifier of the water-right transfer beneficiary (*beneficiary*) and the district of the land where the water has to be transferred (*tr district*). Figure 6 also shows how $F1$ has found a similar domain-case C1 that represents a similar water-right transfer that was granted to $F1$ to promote economy since its land $D_{F1}$ was adjacent (closer than 100 meters) to the land where the water right was offered. Therefore, $F1$ can generate position $pos_{F1}$ which is on the side of $F1$[9] and report this to the other participants of the dialogue with the locution *propose(F1, $pos_{F1}$)*.

In the case of $F2$, the figure shows that it has also retrieved a similar domain-case $C2$, which shows how the same water-right transfer was granted to $F2$ to promote solidarity and irrigate the dry land during a drought. Therefore, $F2$ can generate a position that is on its favour, $pos_{F2}$, and it will communicate this by putting forward the locution *propose(F2, $pos_{F2}$)*.

[9] In this example, we assume that agents only propose the positions that are on their favour.

Once the agents have proposed their positions, the basin administrator *BA* has to decide between them. Therefore, it asks $F1$ and $F2$ to provide an argument to support their positions by using the locutions *why(BA, F1, $pos_{F1}$)* and *why(BA, F2, $pos_{F2}$)*. Assuming that $F1$ and $F2$ are willing to collaborate, they can answer the *BA* with the locutions to put forward the following arguments (in accordance with the structure proposed in (Heras, 2011, Chapter 4)):

Supporting argument of $F1$ (with the locution *assert(F1, BA, SAF1)*):
$SAF1 == \{F1tr, EC, \{Premises, \{C1\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}\}$

Supporting argument of $F2$ (with the locution *assert(F2, BA, SAF2)*):
$SAF2 = \{F2tr, SO, \{Premises, \{C2\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}\}$

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 $F2tr$ mean that the transfer is granted to $F1$ and $F2$, respectively. In accordance with the values of the agents, we assume that the closer the lands are the cheaper the transfers between them are and then $SAF1$ would promote economy. We also assume that crops on dry lands are lost and that helping people to avoid losing crops promotes solidarity. Thus, $SAF2$ would promote solidarity.

Now, the *BA* has to evaluate the arguments of $F1$ and $F2$, attack them if possible, and decide the beneficiary of the water-right transfer. Also, let us assume that, as basin administrator, BA knows an extra premise that states that there is a drought in the basin. First, this new premise matches an argumentation scheme of its ontology, $S1$, which changes the value preference order of the basin in case of drought (such as the argumentation scheme shown in section 3). Thus, this scheme will change the social context of the attack argument that the *BA* is going to create. Since 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.

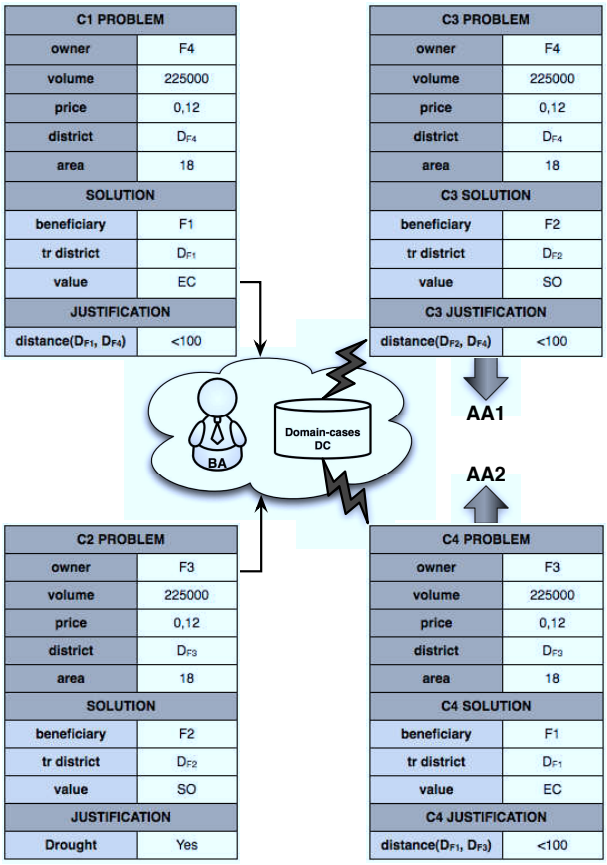


Figure 8: Counter-examples for $C1$ and $C2$

Thus, the *BA* will check its case-base of domain-cases (*DC*) to find counter-examples for $C1$ and $C2$. As shown in Figure 8, 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 by using the locutions:

*attack(BA, F1, AA1)*, where AA1 = {~C1, SO, {Premises ∪ {Drought}, ∅, ∅, S1, ∅, ∅, ∅, {C3}}}

Here, $AA1$ undercuts $SAF1$ by attacking its support element $C1$ with the counter-example $C3$. We assume that by attacking the argument of $F1$, the *BA* supports the argument of $F2$ and then promotes solidarity (SO):

*attack(BA, F2, AA2)*, where AA2 = {~C2, EC, {Premises ∪ {Drought}, ∅, ∅, S1, ∅, ∅, ∅, {C4}}}

$AA2$ undercuts $SAF2$ 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).

Then, the *BA* 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 the extended description and generates an attack argument to $F1$:

*attack(BA, F1, AA3)*, where AA3 = {~C1, SO, {Premises ∪ {Drought}, ∅, ∅, S1, ∅, {Drought}, ∅, ∅}}

In this case, $AA3$ undercuts $SAF1$ by attacking its supporting element $C1$ with the distinguishing premise *drought*. Again, we assume that by 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, 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 $EC <^{S_t}_{RB} J <^{S_t}_{RB} SO$). However, the *BA* still has to decide which argument ($AA1$ or $AA3$) it would select to attack $SAF1$. To do that, *BA* generates an argument-case for each argument and checks its case-base of argument-cases to decide which one is the best argument to pose in view of 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 (such as the one shown in Table 2 of section 3). However, the information of the group that the agents belong to does not match the current information. 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 $F1$ and put forward the locution *attack(BA, F1, AA1)*. Note that if the social context of the argument-case retrieved for $AA3$ had matched the current social context, the basin administrator would have a powerful reason to propose $AA1$ to attack $SAF1$. Also, the *BA* would never propose $AA3$ as an alternative candidate if $AA1$ were rejected.

When F1 receives the attack, it has to evaluate the attack argument in view of its preferences and knowledge resources and the dependency relations of the society. Then, it will realise that $SAF1$ does not defeat $AA1$ from its point of view, since the *BA* has a power dependency relation with every farmer ($Farmer <^{S_t}_{Power}$ *Basin Administrator*). Then, it would try to generate more support for its position. If F1 cannot find such support, it would have to withdraw $pos_F1$ with the locution *noCommit(F1, $pos_{F1}$)*. If no more positions and arguments are provided, the *BA* will close the dialogue and send the locution *accept(BA, all, $pos_{F2}$)*, which grants $F2$ the water-right transfer agreement.

Although the example in this section presents a simple dialogue between agents, it clearly demonstrates how agents' arguments can be managed and interchanged by using our dialogue game protocol. The following section discusses related work.

## 6. Related Work

Dialogue games have been used for multiple purposes in computational linguistics, AI (Bench-Capon, 1998), and philosophy (specifically in argumentation theory (Hamblin, 1970)(MacKenzie, 1979)). In CBR systems, they have been applied to model human reasoning about legal precedents (Prakken and Sartor, 1998). In MAS, their more successful application consists of using them as a tool for the specification of communication protocols between agents. Thus, we can find abundant bibliography that formalises agent interaction protocols by using different dialogue games (Amgoud et al., 2000)(Maudet and Chaib-draa, 2002). Some other examples of dialogue

game protocols about specific types of dialogues are: *information seeking* (Hulstijn, 2000), *persuasion* (Prakken and Sartor, 1998)(Atkinson, 2005)(Wardeh et al., 2008), *negotiation* (Sadri et al., 2001)(Karunatillake et al., 2009), *inquiry* (McBurney and Parsons, 2001), and *deliberation* (McBurney et al., 2007). In contrast, in the protocol presented this work we do not focus on a specific type of dialogue; instead, we have proposed a generic dialogue game that can be used in deliberative, persuasive, or negotiation dialogues where a group of agents must reach an agreement about the solution to apply to a generic problem of any type (e.g., resource allocation, classification, prediction, etc.) that could be described with a set of features. Furthermore, to our knowledge no research has been done to propose a dialogue game that is based on case-based knowledge resources that agents can use to manage agreement processes in agent societies. All of these works rely on rule-based frameworks, with limitted application in open MAS for real domains due to the need of eliciting a previous model of the domain, as explained in section 1.

A particular element of dialogue games, *commitment stores*, has been widely used in the area of MAS. The fact that an agent utters a certain proposition during the dialogue means that this agent incurs a certain level of commitment to this proposition and its implications or, at least, that the agent has certain support to justify this utterance. The concept of commitment stores comes from the study of *fallacies* (poor reasoning patterns that in some way imitate valid reasoning patterns) developed by Hamblin in (Hamblin, 1970). According to this work, formal reasoning systems have public commitment stores for each participant, whose commitments can be withdrawn under certain circumstances. The inclusion of a new commitment gives rise to a previous verification that guarantees the coherence of the information of the store. Following Hamblin's approach, commitments have a purely dialogical processing (he calls them *propositional commitments*) and are associated to beliefs that do not necessary correspond with the actual beliefs of the participant. Furthermore, commitments may not hold outside of the dialogue context. In this work, we use the concept of dialogue games to model the interaction between the agents that belong to a society. In doing so, we assume that the commitments that the agents make during the dialogue are stored in commitment stores that are fully accessible to their owner and partially accessible to the other participants of the dialogue. In this sense, on the contrary to Hamblin's approach, our commitment stores are not completely public in order to preserve the privacy of the arguments interchanged in two-party dialogues between a pair of agents. However, we also endorse the view of Hamblin on the notion of commitments as propositional commitments that agents incur during the dialogue, with no effect once the dialogue is terminated.

Another approach for the concept of commitment was provided by Walton and Krabbe in (Walton and Krabbe, 1995). In this work, commitments are understood as obligations of participants to incur, maintain, or execute a certain course of action (they are *action commitments*). In this case, the commitments made during the dialogue can force the participants to perform certain actions outside of the dialogue context. For these authors, commitments can also represent the fact of uttering statements in the dialogue. Therefore, propositional commitments are viewed as a specific type of action commitments. In our work, we do not consider commitments once the dialogue finishes and the contents of commitment stores are deleted at the end of each dialogue.

Finally, a different approach for commitments was presented by Singh in (Singh, 2000), who proposes a social semantics for agent communication languages. According to Singh, the participants of the dialogue have to express their *social commitments*. These commitments represent public expressions of their mental states, for example their beliefs about certain propositions and their intentions to execute actions in the future, which are relevant to the dialogue. Therefore, by observing these expressions, locutions in the dialogue can be linked to the mental states of agents. In this work, agents have a partial view of the information and locutions conveyed in the dialogue. As pointed out in section 4 each agent has a full view of the commitment store of the other agent engaged with it in a two-party dialogue, but the rest of agents can only see the positions proposed by these agents in the dialogue, but not the arguments that they interchange. This preserves the privacy of the arguments that an agent puts forward in its argumentation dialogue with another agent. In addition, Singh's work assumes that agents are cooperative and honest and do not make expressions to falsely represent its mental states willfully. In our work, we cannot make such assumptions, since they are unrealistic to model open MAS. Our agents are able to make proposals at their convenience and they have to justify them only if requested. In that case, we acknowledge that we do not preclude agents to show false pieces of evidence to support their positions and arguments. Then, we assume that the normative level of the system includes norms to punish such violations of the global good of the society.

Despite the prolific applications of dialogue games in MAS, as discussed by Maudet in (Maudet and Evrard, 1998), a commonly accepted theory of dialogue games that is generic and suitable for any type of dialogue does not yet exist. However, there is a common set of requirements among the models based on dialogue games that defines their *syntax*. In the literature, we can find two main approaches for the syntactic definition of dialogue games. On

the one hand, the work in (McBurney and Parsons, 2002a), which is based on Maudet's requirements, proposes a definition for the components that a dialogue game should have. On the other hand, a different view of the elements of dialogue games is presented in (Prakken and Sartor, 1998). The approach of McBurney and Parsons is prospective (looking forward to model systems that do not yet exist). Opposite to this proposal, Prakken's approach is retrospective (looking back to reconstruct or explain what happened in a dialogue). Therefore, McBurney and Parson's approach can be considered as more suitable for modelling the dialogue between a set of heterogeneous agents whose interactions will determine the dynamics and operation of the system. Therefore, we have followed this approach in our work. By contrast, Prakken's approach assumes a presupposed knowledge about the domain that remains inalterable throughout the dialogue. However, in open MAS, the context can also be changed as new agents enter in the system and new common knowledge is available.

Together with the definition of the syntax, a definition of semantics must be specified to provide a formal definition of the dialogue game. This semantics is concerned with the truth on falsity of utterances. There are different types of semantics for agent communication protocols and dialogue games (van Eijk, 2002). One type of semantics, the *axiomatic* semantics, defines each locution of the protocol in terms of the pre-conditions that must exist before the locution can be uttered and the post-conditions that apply after its utterance. Axiomatic semantics can be *public* or *private* (McBurney, 2002). In the public one, the pre-conditions and post-conditions describe states or conditions of the dialogue that are publicly observable by all its participants, whereas in the private one some pre-conditions or post-conditions describe states or conditions of the dialogue that are only observable by some participants. Another type of semantics is called *operational* semantics. This semantics views the dialogue game protocol as an abstract state machine and precisely defines the transitions between states. The transitions are triggered by the utterance of each locution. The dialogue game proposed in this paper has been formalised by specifying its operational semantics, which provides an intuitive view of the protocol dynamics. Nevertheless, the axiomatic semantics of the protocol has also been defined and can be consulted in (Heras, 2011, Chapter 4).

In a third type of semantics, *denotational* semantics, each element of the language syntax is assigned a relationship to an abstract mathematical entity (its denotation). The *possible worlds* of Kripke (Kripke, 1959) is an example of such a semantics. Finally, there is a specific type of denotational semantics, the *game-theoretic* semantics, where each well-formed statement of the language is associated with a conceptual game between two players, a protagonist and an antagonist. A statement is considered to be true if there is a winning strategy for the protagonist in the associated game (a rule that gives that player moves such that executing them guarantees the player can win the game, no matter what moves are made by the antagonist).

Game-theoretical semantics are usually applied to abstract argumentation frameworks where the strategies of agents determine which argument(s) they will reveal in each argumentation step. However, they assume the existence of a pre-defined utility function about the payoff that an agent obtains for winning the dialogue or having accepted more or fewer arguments. Game theory assumes complete knowledge of the space of arguments proposed in the argumentation framework. There is a large body of literature on mechanism design and game-theoretical models of argumentation (mainly negotiation) in MAS (Rahwan and Reed, 2009). These approaches are typically concerned with the problem of designing mechanisms that provide rewards to individual agents to adopt a certain negotiation strategy. However, opposite to our work, these approaches do not analyse how agents take into account their preferences over values and their dependency relations to manage argumentation dialogues. In addition, game-theoretical assumptions are unrealistic in an argumentation dialogue between heterogeneous agents that have individual and private knowledge resources to generate arguments, which is our case.

## 7. Conclusions

This paper has presented a dialogue game protocol that agents of a case-based argumentation framework can use to interact and engage in argumentation dialogues. The protocol advances research in the investigation of dialectical systems for MAS in the sense that it provides agents with a formalised and structured way of arguing taking into account their social context. The syntax of the protocol has been detailed by defining its locutions, commencement rules, rules for the combination of locutions, commitment rules, rules for the speaker order, and termination rules. The *operational* semantics of the locutions are defined. This semantics views each locution as a transition in an abstract state-machine that represents the possible stages that can be reached during the dialogue.

This work has introduced a running example that motivates the need for a dialogue protocol that controls agreement processes in agent societies and takes into account the social context of agents. A specific dialogue in this scenario has also been presented. This water-right transfer domain is complex enough to be used to illustrate the

performance of the protocol. However, many water-right transfers are usually agreed upon by the water users, without any recording of the terms and outcome of the agreement. Therefore, due to this fact and due to restrictive privacy laws to access this type of data, the actual implementation of the system in this domain still remains to be done in future work. Nevertheless, during this project we have elicited the knowledge of experts from the water market domain to design the protocol. In addition, our framework has been implemented as an argumentation API in the *Magentix2* agent platform and in (Heras et al., 2013), we run several experiments to evaluate our argumentation framework by simulating an agreement process in a real domain.

In this work we have assumed that a proponent agent addresses its arguments to an opponent of its same group, having complete knowledge of the opponents' social context. However, in real systems, some features of argument-cases could be unknown. For instance, the proponent of an argument obviously knows its value preferences and probably knows the preferences of its group, however, in a real open MAS, it is unlikely that the opponent's value preferences are known. Nevertheless, the proponent might 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 may be unknown, but the proponent could use its experience with other agents of the opponent's group and infer them. Therefore, many interesting questions on how to infer the opponents' social context remain to be studied as future work. A battery of tests to evaluate the influence of the knowledge that an agent has about the social context of its opponents on the performance of the system was developed and analysed in (Heras, 2011, Chapter 6). Even though the framework is flexible enough to cope with this lack of knowledge, the reliability of the conclusions drawn from previous experience would not be as good.

Furthermore, the features of the proponent or the opponent 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 addition, the argumentation dialogue is centralised by the basin administrator and agents do not speak to each other directly; however the basin administrator could use the information provided by an agent to attack the arguments of another agent. Nevertheless, our protocol is conceived to serve for both mediated and face-to-face argumentation dialogues.

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. Tackling doing strategies in argumentation dialogues is a complex problem that we are dealing with in current research. For instance, to improve efficiency in a negotiation an argumentation dialogue could be terminated if it were similar to a previous one that didn't reach an agreement. Otherwise, opponent moves in a dialogue could be inferred by looking at a similar previous dialogue with the same opponent.

## Acknowledgements

This work is supported by the Spanish government grants CONSOLIDER INGENIO 2010 CSD2007-00022, MINECO/FEDER TIN2012-36586-C03-01, and TIN2011-27652-C03-01.

## References

Amgoud, L., Maudet, N., Parsons, S., 2000. Modelling dialogues using argumentation, in: 4th International Conference on MultiAgent Systems, ICMAS-00, IEEE Press.

Atkinson, K., 2005. What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning. Ph.D. thesis. Liverpool University.

Augusto, J., Simari, G., 2001. Temporal Defeasible Reasoning. Knowledge and Information Systems 3, 287–318.

Baader, F., Horrocks, I., Sattler, U., 2007. Handbook of Knowledge Representation. Elsevier. chapter Description Logics. pp. 135–179.

Bench-Capon, T., Sartor, G., 2003. A Model of Legal Reasoning with Cases Incorporating Theories and Values. Artificial Intelligence 150, 97–143.

Bench-Capon, T.J., 1998. Specification and Implementation of Toulmin Dialogue Game, in: International Conferences on Legal Knowledge and Information Systems, JURIX-98, IOS Press. pp. 5–20.

Botti, V., Garrido, A., Gimeno, J.A., Giret, A., Igual, F., Noriega, P., 2010. An Electronic Institution for Simulating Water-Right Markets, in: 3rd Workshop on Agreement Technologies, WAT-10, pp. 3–18.

Botti, V., Garrido, A., Giret, A., Igual, F., Noriega, P., 2009a. On the design of mWater: a case study for Agreement Technologies, in: 7th European Workshop on Multi-Agent Systems - EUMAS-09.

Botti, V., Garrido, A., Giret, A., Noriega, P., 2009b. 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, Springer. pp. 1–10.

Carrascosa, C., Rebollo, M., 2009. Agreement Spaces for Counselor Agents, in: 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS-09, ACM Press. pp. 1205–1206.

Dignum, F., Weigand, H., 1995. Communication and Deontic Logic, in: Wieringa, R., Feenstra, R. (Eds.), Information Systems - Correctness and Reusability. Selected papers from the IS-CORE Workshop, World Scientific Publishing Co.. pp. 242–260.

van Eijk, R.M., 2002. Semantics of Agent Communication: An Introduction, in: Foundations and Applications of Multi-Agent Systems, UKMAS 1996-2000, Selected Papers, Springer-Verlag. pp. 152–168.
Garrido, A., Giret, A., Noriega, P., 2009. mWater: a Sandbox for Agreement Technologies, in: 12th International Congress of the Catalan Association of Artificial Intelligence - CCIA-09, IOS Press. pp. 252–261.
Giret, A., Garrido, A., Botti, V., 2010. D8.2.1 Report. mWater. Technical Report AT/2008/D8.2.1/v0.1. Universidad Politecnica de Valencia.
Hamblin, C.L., 1970. Fallacies. Methuen and Co. Ltd.
Heras, S., 2011. Case-Based Argumentation Framework for Agent Societies. Ph.D. thesis. Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València. http://hdl.handle.net/10251/12497.
Heras, S., Jordán, J., Botti, V., Julián, V., 2013. Argue to Agree: a Case-Based Argumentation Approach. International Journal of Approximate Reasoning 54, 82–108.
Honey-Roses, J., 2007. Assessing the potential of water trading in Spain. ENR 319 Advanced International Environmental Economics. Prof. T. Panayotou at Harvard's John F. Kennedy School of Government.
Horrocks, I., Patel-Schneider, P., 2004. Reducing OWL entailment to description logic satisfiability. Journal of Web Semantics 1, 345–357.
Huhns, M.N., Singh, M.P., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, L., Goradia, H., Jennings, N., Lakkaraju, K., Nakashima, H., Parunak, H.V.D., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavala, L., 2005. Research Directions for Service-Oriented Multiagent Systems. IEEE Internet Computing 9, 65–70.
Hulstijn, J., 2000. Dialogue Models for Inquiry and Transaction. Ph.D. thesis. University of Twente.
Jurisica, I., Mylopoulos, J., Yu, E., 2004. Ontologies for Knowledge Management: An Information Systems Perspective. Knowledge and Information Systems 6, 380–401.
Karunatillake, N.C., Jennings, N.R., Rahwan, I., McBurney, P., 2009. Dialogue Games that Agents Play within a Society. Artificial Intelligence 173, 935–981.
Kripke, S., 1959. A completeness proof in modal logic. Journal of Symbolic Logic 24, 1–14.
Luck, M., McBurney, P., 2008. Computing as interaction: agent and agreement technologies, in: IEEE International Conference on Distributed Human-Machine Systems, IEEE Press.
MacKenzie, J.D., 1979. Question-begging in non-cumulative systems. Philosophical Logic 8, 117–133.
Maudet, N., Chaib-draa, B., 2002. Commitment-based and Dialogue-game based Protocols-News Trends in Agent Communication Language. Knowledge Engineering Review 17, 157–179.
Maudet, N., Evrard, F., 1998. A generic framework for dialogue game implementation, in: 2nd Workshop on Formal Semantics and Pragmatics of Dialogue, University of Twente. pp. 185–198.
McBurney, P., 2002. Rational Interaction. Ph.D. thesis. Department of Computer Science, University of Liverpool, Liverpool, UK.
McBurney, P., Hitchcock, D., Parsons, S., 2007. The eightfold way of deliberation dialogue. International Journal of Intelligent Systems 22, 95–132.
McBurney, P., Parsons, S., 2001. Representing epistemic uncertainty by means of dialectical argumentation. Annals of Mathematics and Artificial Intelligence, Special Issue on Representations of Uncertainty 32, 125–169.
McBurney, P., Parsons, S., 2002a. Dialogue Games in Multi-Agent Systems. Informal Logic. Special Issue on Applications of Argumentation in Computer Science 22, 257–274.
McBurney, P., Parsons, S., 2002b. Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information 11, 315–334.
McBurney, P., Parsons, S., 2004. Locutions for argumentation in agent interaction protocols, in: Revised Proceedings of the International Workshop on Agent Communication, AC-04, Springer. pp. 209–225.
McBurney, P., Parsons, S., 2009. Argumentation in Artificial Intelligence. Springer. chapter Dialogue games for agent argumentation. pp. 261–280.
McBurney, P., Parsons, S., Wooldridge, M., 2002. Desiderata for agent argumentation protocols, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS-02, ACM Press. pp. 402–409.
Ossowski, S. (Ed.), 2013. Agreement Technologies. volume 8. Springer.
Panayotou, T., 2007. Environment and Natural Resources 319. Advanced International Environmental Economics. Lecture 21: Issues in the Economics and Management of Water Resources. Kennedy School of Government, Harvard University.
Prakken, H., 2006. Formal systems for persuasion dialogue. The Knowledge Engineering Review 21, 163–188.
Prakken, H., 2010. An abstract framework for argumentation with structured arguments. Argument and Computation 1, 93–124.
Prakken, H., Reed, C., Walton, D., 2005. Dialogues about the burden of proof, in: Proceedings of the 10th International Conference on Artificial Intelligence and Law, ICAIL-05, ACM Press. pp. 115–124.
Prakken, H., Sartor, G., 1998. Modelling reasoning with precedents in a formal dialogue game. Artificial Intelligence and Law 6, 231–287.
Rahwan, I., Banihashemi, B., Reed, C., Walton, D., Abdallah, S., 2011. Representing and Classifying Arguments on the Semantic Web. The Knowledge Engineering Review 26, 487–511.
Rahwan, I., Reed, C., 2009. Argumentation in Artificial Intelligence. Springer. chapter The Argument Interchange Format. pp. 383–402.
Sadri, F., Toni, F., Torroni, P., 2001. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences, in: Revised Papers from the 8th International Workshop on Intelligent Agents VIII, ATAL-01, Springer. pp. 405–421.
Schneider, S. (Ed.), 1996. Encyclopedia of Climate and Weather. Oxford University Press.
Shoham, Y., Leyton-Brown, K., 2009. Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations. Cambridge University Press.
Singh, M., 2000. A social semantics for agent communication languages, Springer. pp. 31–45.
Tennent, R.D., 1991. Semantics of Programming Languages. Prentice Hall.
del Val, E., Rebollo, M., Botti, V., 2014. Enhancing decentralized service discovery in open service-oriented multi-agent systems. Autonomous Agents and Multi-Agent Systems 28, 1–30.
Verheij, B., 2009. Argumentation in Artificial Intelligence. Springer. chapter The Toulmin Argument Model in Artificial Intelligence. pp. 219–238.
Walton, D., Krabbe, E.C.W., 1995. Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press.
Walton, D., Reed, C., Macagno, F., 2008. Argumentation Schemes. Cambridge University Press.
Wardeh, M., Bench-Capon, T., Coenen, F.P., 2008. PISA - Pooling Information from Several Agents: Multiplayer Argumentation From

Experience, in: Proceedings of the 28th SGAI International Conference on Artificial Intelligence, AI-2008, Springer. pp. 133–146.