UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

PhD THESIS

CASE-BASED ARGUMENTATION IN AGENT SOCIETIES

Author: Stella Heras Barberá
Supervisors: Dr. Vicente J. Julián Inglada
Dr. Vicente J. Botti Navarro

PROGRAMA DE DOCTORADO EN INFORMÁTICA

Valencia
18.05.2011

Resumen

Hoy en día los sistemas informáticos complejos se pueden ven en términos de los servicios que ofrecen y las entidades que interactúan para proporcionar o consumir dichos servicios. Los sistemas multi-agente abiertos, donde los agentes pueden entrar o salir del sistema, interactuar y formar grupos (coaliciones de agentes u organizaciones) de forma dinámica para resolver problemas, han sido propuestos como una tecnología adecuada para implementar este nuevo paradigma informático. Sin embargo, el amplio dinamismo de estos sistemas requiere que los agentes tengan una forma de armonizar los conflictos que surgen cuando tienen que colaborar y coordinar sus actividades. En estas situaciones, los agentes necesitan un mecanismo para argumentar de forma eficiente (persuadir a otros agentes para que acepten sus puntos de vista, negociar los términos de un contrato, etc.) y poder llegar a acuerdos.

La argumentación es un medio natural y efectivo para abordar los conflictos y contradicciones del conocimiento. Participando en diálogos argumentativos, los agentes pueden llegar a acuerdos con otros agentes. En un sistema multi-agente abierto, los agentes pueden formar sociedades que los vinculan a través de relaciones de dependencia. Estas relaciones pueden surgir de sus interacciones o estar predefinidas por el sistema. Además, los agentes pueden tener un conjunto de valores individuales o sociales, heredados de los grupos a los que pertenecen, que quieren promocionar. Las dependencias entre los agentes y los grupos a los que pertenecen y los valores individuales y sociales definen el contexto social del agente. Este contexto tiene una influencia decisiva en la forma en que un agente puede argumentar y llegar a acuerdos con otros agentes. Por tanto, el contexto social de los agentes debería tener una influencia decisiva en la representación computacional de sus argumentos y en el proceso de gestión de argumentos.

La principal contribución de esta tesis es la propuesta de un marco para la argumentación en sociedades de agentes que hace uso de la metodología de razonamiento basado en casos. El razonamiento basado en casos es especialmente conveniente cuando se tiene poco conocimiento a priori sobre el dominio de aplicación, pero sin embargo, en la práctica se puede disponer fácilmente de casos o ejemplos. La mayoría de los sistemas argumentativos producen argumentos mediante la aplicación de un conjunto de reglas de inferencia. El dinamismo de los sistemas multi-agente hace que sea difícil definir previamente el conjunto de reglas que modelan el comportamiento de los agentes. Por tanto, mientras que el razonamiento con un conjunto predefinido de reglas puede ser complejo, el seguimiento de los argumentos que los agentes proponen durante los diálogos argumentativos es relativamente simple. El marco propuesto permite a los agentes representar argumentos y razonar sobre ellos, teniendo en cuenta su contexto social en la forma en la que los agentes pueden dialogar. Así, las dependencias sociales entre los agentes y sus valores individuales y sociales también son consideradas. Además, los agentes que soportan este marco pueden participar en diálogos de argumentación siguiendo estrategias de diálogo diferentes. Con estas estrategias, los agentes pueden seleccionar en cada momento el argumento más adecuado para alcanzar sus objetivos.

El marco de argumentación propuesto proporciona a los agentes los recursos de necesarios para generar posiciones y argumentos de forma individual. Por un lado, los agentes disponen de una base de casos de dominio, con casos de dominio que representan problemas anteriores y las soluciones que fueron aplicadas para resolverlos. Por otra parte, los agentes disponen de una base de casos de argumentos, con casos argumento que representan experiencias de argumentación previas y su resultado final. Además, los agentes pueden acceder a un conjunto de esquemas de argumentación, que representan patrones estereotipados de razonamiento común en el dominio de aplicación donde se implementa el marco. Todos estos recursos están representados mediante el uso de ontologías. Por lo tanto, en esta tesis se ha desarrollado una ontología de argumentación basada en casos que actúa como lenguaje de representación para los recursos de conocimiento propuestos. Finalmente, se ha definido el proceso de razonamiento que los agentes de nuestro marco pueden utilizar para generar posiciones y argumentos.

Para permitir a los agentes interactuar y controlar el proceso de argumentación entre ellos, se ha desarrollado un protocolo de juego de diálogo. Además, se ha propuesto un conjunto de estrategias de diálogo que los agentes pueden utilizar para mejorar el rendimiento de los diálogos en los que participan.

Por último, nuestra propuesta ha sido evaluada en dos casos de estudio. Por un lado, la especificación formal del marco ha sido aplicada en el dominio de un sistema de transferencia de derechos de agua, donde los agentes participan en diálogos argumentativos para llegar a acuerdos sobre la asignación de recursos hídricos. Este es un ejemplo teórico donde las propiedades semánticas del marco han sido validadas. Por otro lado, el marco se ha implementado en un sistema de atención al cliente. Este caso consiste en un servicio de asistencia implementado en un centro de llamadas en el que varios operadores deben llegar a un acuerdo para resolver las incidencias recibidas por el centro.

Resum

Actualment els sistemes informàtics complexos es poden veure en termes dels serveis que ofereixen i les entitats que interactuen per proporcionar o consumir aquests serveis. Els sistemes multi-agent oberts, on els agents poden entrar o sortir del sistema, interactuar i formar grups (coalicions d'agents o organitzacions) de manera dinàmica per a resoldre problemes, han estat proposats com una tecnologia adequada per implementar aquest nou paradigma informàtic. No obstant això, l'ampli dinamisme d'aquests sistemes requereix que els agents tinguen una forma d'harmonitzar els conflictes que sorgeixen quan han de col.laborar i coordinar les seues activitats. En aquestes situacions, els agents necessiten un mecanisme per argumentar de manera eficient (persuadir altres agents perquè accepten els seus punts de vista, negociar els termes d'un contracte, etc.) i poder arribar a acords.

L'argumentació és un medi natural i efectiu per abordar els conflictes i contradiccions del coneixement. Participant en diàlegs argumentatius, els agents poden arribar a acords amb altres agents. En un sistema multi-agent obert, els agents poden formar societats que els vinculen a través de relacions de dependència. Aquestes relacions poden sorgir de les seves interaccions o estar predefinides pel sistema. A més a més, el agents poden tindre una serie de valors individuals o socials, heretats dels grups als que pertanyen, que volen promoure. Les dependències entre els agents i els grups als quals pertanyen i el els valors individuals i socials defineixen el context social de l'agent. Aquest context té una influència decisiva en la forma en què un agent pot argumentar i arribar a acords amb altres agents. Per tant, el context social dels agents hauria de tenir una influència decisiva en la representació computacional dels seus arguments i en el procés de gestió d'arguments.

La principal contribució d'aquesta tesi és la proposta d'un marc per l'argumentació en societats d'agents que utilitza la metodologia d'argumentació basada en casos. El raonament basat en casos és especialment convenient quan es té poc coneixement a priori sobre el domini d'aplicació, però no obstant això, en la pràctica es pot disposar fàcilment de casos o exemples. La majoria dels sistemes argumentatius produeixen arguments mitjançant l'aplicació d'un conjunt de regles d'inferència. El dinamisme dels sistemes multi-agent fa que sigue difícil definir prèviament el conjunt de regles que modelen el comportament dels agents. Per tant, mentre que el raonament amb un conjunt predefinit de regles pot ser complex, el seguiment dels arguments que els agents proposen durant els diàlegs argumentatius és relativament simple. El marc proposat permet als agents representar arguments i raonar sobre ells, tenint en compte el seu context social en la forma en què els agents poden dialogar. Així, les dependències socials entre els agents i els seus valors individuals i socials també es consideren. A més a més, els agents que suporten aquest marc poden participar en diàlegs d'argumentació seguint estratègies de diàleg diferents. Amb aquestes estratègies, els agents poden seleccionar en cada moment l'argument més adequat per assolir els seus objectius.

El marc d'argumentació proposat proporciona als agents els recursos de coneixement necessaris per generar posicions i arguments de forma individual. D'una banda, els agents disposen d'una base de casos de domini, amb casos de domini que representen problemes anteriors i les solucions que van ser aplicades per resoldre'ls. D'altra banda, els agents disposen d'una base de casos d'arguments, amb casos argument que representen experiències d'argumentació prèvies i el seu resultat final. A més a més, els agents poden accedir a un conjunt d'esquemes d'argumentació, que representen patrons estereotipats de raonament comú en el domini d'aplicació on s'implementa el marc. Tots aquests recursos estan representats mitjançant l'ús d'ontologies. Per tant, en aquesta tesi s'ha desenvolupat una ontologia d'argumentació basada en casos que actua com a llenguatge de representació per als recursos de coneixement proposats. Finalment, s'ha definit el procés de raonament que els agents del nostre marc poden utilitzar per generar posicions i arguments.

Per a permetre als agents interactuar i controlar el procés d'argumentació entre ells, s'ha desenvolupat un protocol de joc de diàleg. A més a més, s'ha proposat un conjunt d'estrategies de diàleg que els agents poden utilitzar per a millorar el rendiment dels diàlegs en què participen.

Finalment, la nostra proposta ha estat avaluada en dos casos d'estudi. D'una banda,

l'especificació formal del marc ha sigut aplicada en el domini d'un sistema de transferència de drets d'aigua, on els agents participen en diàlegs argumentatius per arribar a acords sobre l'assignació de recursos hídrics. Aquest és un exemple teòric on les propietats semàntiques del marc han estat validades. D'altra banda, el marc s'ha implementat en un sistema d'atenció al client. Aquest cas consisteix en un servei d'assistència implementat en un centre de trucades en el qual diversos operadors han d'arribar a un acord per resoldre les incidències rebudes pel centre.

Summary

Nowadays large systems are viewed in terms of the services that they offer and the entities that interact to provide or consume these services. Open multi-agent systems, where agents can enter or leave the system, interact and dynamically form groups (e.g. agents' coalitions or organisations) to solve problems, seems a suitable technology to implement this new computing paradigm. However, the high dynamism of open multi-agent systems requires agents to have a way of harmonising the conflicts that come out when they have to collaborate or coordinate their activities. In those situations, agents need a mechanism to argue (persuade other agents to accept their points of view, negotiating the terms of a contract, etc.) and reach agreements.

Argumentation provides a fruitful means of dealing with conflicts and knowledge inconsistencies. Agents can reach agreements by engaging in argumentation dialogues with their opponents in a discussion. In addition, agents in open multi-agent systems can form societies that link them via dependency relations. These relations can emerge from agents' interactions or be predefined by the system. In addition, agents can have individual and social values, inherited from the groups that they belong, which they want to promote. The dependencies between agents and the group(s) that they belong and the individual and social values define the agents' social context. This context has an important influence in the way agents can reach agreements with other agents. Therefore, agents' social context should have a decisive influence in the computational representation of arguments and in the argument management process.

The main contribution of this PhD work is the proposal of an argumentation framework for agent societies, based on the case-based reasoning methodology. Reasoning with cases is specially suitable when there is a weak (or even unknown) domain theory,

but acquiring examples encountered in practice is easy. Most argumentation systems produce arguments by applying a set of inference rules. In multi-agent systems, the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. Thus, reasoning with a predefined set of rules can be difficult while tracking the arguments that agents put forward in argumentation dialogues could be relatively simple. The framework proposed allows agents to computationally represent arguments and reason about them, taking into account the agents' social context in the way agents can argue. Thus, social dependencies between agents and their individual and social values are also considered. Also, agents that comply with this framework are able to engage in argumentation dialogues following different dialogue strategies. With these strategies, agents can select the most appropriate argument to bring about their desired outcome of the dialogue.

In addition, the framework proposed provides agents with individual knowledge resources to generate their positions and arguments. On one hand, agents have a domain-cases case-base, with domain-cases that represent previous problems and their solutions. On the other hand, agents have an argument-cases case-base, with argument-cases that represent previous argumentation experiences and their final outcome. In addition, agents can accede to a set of argumentation schemes, which represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. All these resources are represented by using ontologies. Thus, we have developed the case-based argumentation ontology that acts as representation language for the knowledge resources proposed. The reasoning process that agents of our framework can use to generate positions and arguments is also defined.

To allow agents to interact and control the argumentation process between them, we have designed a dialogue game protocol. Furthermore, we have proposed a set of dialogue strategies that agents can use to improve the performance of their dialogues.

Finally, we have tested our proposals with two study cases. On one hand, the formal specification of the framework has been applied to a water-right transfer domain where agents engage in argumentation dialogues to reach agreements over the allocation of water resources. This is a theoretic example where the semantic properties of the framework have been validated. On the other hand, the framework has been implemented to develop an application on the customer support domain. Here, we consider a call centre where several operators must reach an agreement to solve the incidences received by the centre.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter

1

Introduction

1.1 Motivation

Nowadays large systems are viewed in terms of the services that they offer and the entities that interact to provide or consume these services. This design paradigm is known as computing as interaction [Luck and McBurney, 2008]. Open Multi-Agent Systems (MAS), where agents can enter or leave the system, interact and dynamically form groups (e.g. agent' coalitions or organisations) to solve problems, seems a suitable technology to implement this new paradigm. However, the high dynamism of open MAS requires agents to have a way of harmonising the conflicts that come out when they have to collaborate or coordinate their activities. In those situations, agents need a mechanism to argue (persuade other agents to accept their points of view, negotiating the terms of a contract, etc.) and reach agreements. In addition, agents in open MAS can form societies that link them via dependency relations. These relations can emerge from agents' interactions or be predefined by the system. Anyway, the dependencies between agents and the group(s) to which they belong define the agents' social context. This context has an important influence in the way agents can reach agreements with other agents (e.g. subordinates are not as willing to accept tasks from equals as they are from superiors or workers do not behave in the same way if they are negotiating their own salaries than if they act as representatives of their trade unions).

Argumentation provides a fruitful means of dealing with conflicts and knowledge inconsistencies. Agents can reach agreements by engaging in argumentation dialogues with their opponents in a discussion. This has made Artificial Intelligence (AI) researchers to pay their attention on argumentation theory [Rahwan and Simari, 2009]. However, most works in the area assume human users interacting with software tools, such as the approaches for argument authoring and diagramming [Rahwan et al., 2007b], OVA[1]. Others focus on defining and analysing the properties of abstract argumentation frameworks [Rahwan and Simari, 2009, Part I]. Most developments in argumentation theory are motivated by the works on case-based legal argumentation [Skalak and Rissland, 1992]. However, many case-based computational frameworks for legal reasoning assume humans interacting with the system or agents that have complete knowledge about the domain (represented in the form of cases that store knowledge about past legal disputes) and play two party dialogues [Bench-Capon and Sartor, 2003]. Hence, the notions of argument and argumentation resources of these systems are not conceived for being acceded only by software agents that perform automatic reasoning processes over them and have individual and partial knowledge about the domain. In MAS, researchers have recently studied argumentation as a mechanism for managing dialogues between agents and reaching agreements [Rahwan and Simari, 2009, Part III]. However, most works on argumentation in MAS take a narrow view on the argument structure [Reed and Grasso, 2007] or use domain-dependent structures for the computational representation of arguments. The few current approaches for case-based argumentation in MAS suffer from this domain-dependency or centralise the argumentation abilities in a *mediator* agent [Heras et al., 2009b]. Therefore, a research challenge on argumentation in MAS is to develop a generic computational representation of arguments that allow autonomous agents to perform argumentation dialogues in different domains. In addition, agents should be able to follow a strategy in the dialogue and decide in each situation which particular utterance is the best to bring about a desired outcome (e.g. to persuade an agent to accept an argument, to win a negotiation, etc.).

Moreover, little work has been done to study the effect of the social context of agents in the way that they argue and manage arguments. Commonly, the term *agent society* is used in the argumentation and AI literature as a synonym for an *agent organisation* [Ferber et al., 2004] or a *group of agents* that play specific roles, follow some interaction patterns and collaborate to reach global objectives [Oliva et al., 2008]. In addition to the dependency relations between agents, they can also have *values*. These values can

[1] `OVA at ARG:dundee: www.arg.dundee.ac.uk`

be individual values that agents want to promote or demote (e.g. solidarity, peace, etc.) or also *social values* inherited from the agents' dependency relations (e.g. in a negotiation, a superior could impose their values to his subordinates or, on the opposite, a trade unionist might have to adopt the values of the collective that he represents). Thus, we endorse the view of value-based argumentation frameworks [Bench-Capon and Atkinson, 2009], which stress the importance of the audience (other participants in the dialogue) in determining whether an argument is persuasive or not. Therefore, we also consider values as an important element of the social context of agents.

Starting from the idea that the social context of agents determines the way in which agents can argue and reach agreements, this context should have a decisive influence in the computational representation of arguments, in the argument management process and in the way agents develop strategies to argue with other agents. While work on argument evaluation and generation has received much attention, the strategic use of arguments has received little attention in the literature [Rahwan, 2006]. Moreover, to our knowledge, few research is done to adapt multi-agent argumentation frameworks to represent and automatically manage arguments of agents taking into account their social context. This opens a new research challenge for the development of a computational framework for design and implementation of MAS in which the participating software agents are able to manage and exchange arguments between themselves taking into account the agents' social context. To deal with this challenge the reasoning process by which agents can automatically generate, select and evaluate arguments in an agent society must be specified. Also, an interesting feature would be to allow agents to learn from argumentation experiences and in this way, make easier to develop dialogue strategies that help them to reach their objectives in the argumentation dialogue. To follow a Case-Based Reasoning (CBR) methodology [Aamodt and Plaza, 1994] could be appropriate to develop the argument management process. Reasoning with cases is specially suitable when there is a weak (or even unknown) domain theory, but acquiring examples encountered in practice is easy. Most argumentation systems produce arguments by applying a set of inference rules. In MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. Thus, reasoning with a predefined set of rules can be difficult. However, tracking the arguments that agents put forward in argumentation dialogues could be relatively simple.

The discussion above raises several questions that this research is intended to answer:

Q1 Are the current approaches in case-based argumentation for MAS suitable to be

applied to open MAS where the agents belong to a society?

Q2 How the argumentation framework proposed in this research can be formalised and its properties analysed?

Q3 How agents in agent societies can automatically manage argumentation processes?

Q4 How agents can learn from argumentation experiences?

Q5 How agents in agent societies can exchange arguments to held efficient argumentation dialogues?

Q6 How agents can follow dialogue strategies during the argumentation process?

Q7 What is the best technology to apply in this context?

Q8 How the advantages of applying the framework can be tested and analysed?

1.2 Objectives

With the aim of providing answers to the questions posed in the previous section, the objective of this PhD work is to propose a framework for case-based argumentation in multi-agent societies. This framework will allow agents to computationally represent arguments and reason about them. The framework considers the agents' social context in the way agents can argue. Thus, social dependencies between agents and their individual and social values are also considered. Also, agents implementing this framework will be able to engage in argumentation dialogues following different dialogue strategies. With these strategies, agents will be able to select the most appropriate argument to bring about their desired outcome of the dialogue. According to its main objectives, the contributions of this work are organised on different levels:

1. On the **State of the Art Revision Level**, related works have been reviewed and analysed to point out their main handicaps to be applied in our agent societies scenario. This analysis would provide the answer to question 1.

2. On the theoretical and **Formal Level**, an abstract argumentation framework that allow agents to argue and improve their argumentation skills is proposed. After that, this framework is instantiated by defining a specific structure for arguments. Question 2 is answered in this level.

3. On the **Agent Level**, our main aim is to provide agents of agent societies with the ability of generating arguments, selecting the best ones to put forward and evaluating incoming arguments and the argumentation process itself. This level will cope with questions 3 and 4. Here, the first step to design MAS whose agents are able to perform argumentation processes is to decide how agents represent and store arguments. There are some requirements that should be met to make a suitable choice for the structure to represent arguments in our social environment. Summarising, this structure should:

   - be computationally tractable and designed to ease the performance of automatic reasoning processes over it;
   - be rich enough to represent knowledge about the domain and social information about agents and their groups;
   - be generic enough to represent different types of arguments and
   - comply with the technological standards of data and argument interchange on the web.

   A knowledge-intensive case-based structure to represent arguments could suit these requirements, since it can be easily interpreted by machines and has highly expressive formal semantics to define complex concepts and relations over them [Diaz-Agudo and Gonzalez-Calero, 2007]. In addition, the reasoning process that agents perform to generate, select and evaluate arguments taking into account their social context is also studied.

4. On the **System Level**, the objective of this thesis is to develop a dialogue game protocol to allow agents in agent societies to engage in argumentative dialogues. This level is related with question 5, 6 and 7. After reviewing current approaches, we have decided to follow a dialogue game approach that could allow agents to use different argumentation resources in the dialogue, such as *distinguishing premises*, *counter-examples* and *argumentation schemes* [Bench-Capon and Sartor, 2003; Walton et al., 2008]. Also, we propose several dialogue strategies that the agents can follow to select the best utterance to bring out in each step of the dialogue.

5. Finally, on the **Evaluation Level**, the hypothesis and proposals of the thesis will be implemented and tested in different cases of study: a social network of recommender agents and a system for the water-right transfer management in a real Spanish river basin. Thus, this level will provide an answer for question 8.

1.3 Structure of the document

This chapter has introduced the PhD proposal, outlining the motivation and objectives of the thesis. The rest of the document is structured as follows.

Chapter 2 introduces the uses of argumentation theory in AI and explains the main concepts of argumentation theory and its applications to AI. This chapter also reviews the main research approaches in CBR-based argumentation in MAS and points out some open research issues in the area.

Chapter 3 introduces the case-based argumentation framework proposed, explaining its knowledge resources. The chapter shows first the abstract framework and then, instantiates it, defining the structure of its elements. Furthermore, it presents the reasoning process that agents can follow to generate, select and evaluate arguments.

Chapter 4 specifies the dialogue protocol that can use the agents of an agent society to exchange arguments and engage in an argumentation process to reach agreements. The chapter presents the syntax and semantics of the protocol. In addition, several dialogue strategies that depend on the agents' profiles and preferences are devised.

In Chapter 5, an example application of the framework in the domain of water-rights transfer in a river basin is provided. This is a theoretic example that illustrates the type of environment and problem that the framework is able to deal with and analyses its properties from different perspectives.

Chapter 6 presents an implementation of the framework in the domain of a helpdesk application, where a group of operators must solve an incidence reported by a user. Here, the framework is empirically evaluated, testing its performance under different evaluation criteria.

Chapter 7 summarises the main contributions of this thesis and proposes future work on this area. Finally, the bibliographical work published during the development of this PhD thesis is referenced.

Part II

State of the Art

Chapter

2

State of the Art

2.1 Introduction

As pointed out in Chapter 1, all along the history research done on argumentation on Artificial Intelligence (AI) have experienced mutual contributions. The argumentation theory has produced important benefits on many AI research areas, from its first uses as an alternative to formal logic for reasoning with incomplete and uncertain information to its more recent applications in Multi-Agent Systems (MAS) [Bench-Capon and Dunne, 2007; Rahwan and Simari, 2009]. Currently, the study of argumentation in this area has gained a growing interest. The reason behind is that having argumentation skills increases the agents' autonomy and provides them with a more intelligent behaviour.

An autonomous agent should be able to act and reason as an individual entity on the basis of its mental state (beliefs, desires, intentions, goals, etc.). As member of

a MAS, an agent interacts with other agents whose goals could come into conflict with those of the agent. Moreover, if a dynamic and open MAS is considered, the knowledge that an agent has about the environment, its neighbours and its mental state can change in the course of time. In addition, agents can have a social context that imposes dependency relations between them and preference orders among a set of potential values to promote/demote. Therefore, agents must have the ability of reaching agreements that harmonise their mental states and that solve their conflicts with other agents by taking into account their social context and values. Argumentation is a natural way of reaching agreements between several parties with opposing positions about a particular issue. The argumentation techniques, hence, can be used to facilitate the agents' autonomous reasoning and to specify interaction protocols between them [Rahwan, 2006].

Case-Based Reasoning (CBR) [Aamodt and Plaza, 1994] is another research area where the argumentation theory has produced a wide history of successful applications. According to the CBR methodology, a new problem can be solved by searching in a case-base for similar precedents and adapting their solutions to fit the current problem. This reasoning methodology has a high resemblance with the way by which people argue about their positions, trying to justify them on the basis of past experiences. The argumentation theory concepts and techniques have been successfully applied in a great number of CBR systems, specially in those that work in legal domains, where a plaintiff and a defendant argue over their opposing positions in court. In this case, if *'common law'* is the legal system that applies, similar cases should be resolved with similar verdicts.

The work done in the eighties about legal CBR fostered the argumentation research in the AI community [Rissland et al., 2006]. From then on, the good results of CBR systems in argumentation domains suggest that this type of reasoning is suitable to manage argumentation processes. Nowadays, MAS research community is endeavouring to broaden the applications of the paradigm to more real environments, where heterogeneous agents could enter in (or leave) the system, form societies and interact with other agents [Ossowski et al., 2007]. Also, MAS have been proposed as a suitable technology to implement the new paradigm of computing as interaction [Luck and McBurney, 2008], where large systems can be viewed or designed in terms of the services they offer and the entities that interact to provide or consume these services. As it was pointed out before, the high dynamism of these *open MAS* gives rise to a greater need for a way of reaching and managing agreements that harmonise conflicts.

Moreover, this type of systems also poses other potential problems to overcome. Common assumptions about the agents of most MAS, such as honesty, cooperativeness and trustworthiness cannot be longer taken as valid hypothesis in open MAS. Therefore, there is an obvious need for providing the agents of an open MAS with individual reasoning and learning capabilities that make them more intelligent and autonomous and prevent them from the potential attacks of interested agents.

Our aim with this state of the art review is to study the feasibility of using CBR as the main reasoning and learning method in a framework that allows the agents of an open MAS to reach agreements via argumentation and to learn by experience to perform a more effective (strategic) dialogue. We follow a CBR approach since this methodology is very suitable in domains with weak or unknown domain theory, where defining the set of rules that represent the behaviour of the agents is infeasible, but acquiring examples encountered in practice is easy. With this purpose, we have reviewed the approaches of hybrid case-based MAS for argumentation that have been proposed in the literature. As a result of this work, the current CBR contributions to argumentation in MAS have been identified. In addition, research challenges in the area have been specified by pointing out several open research issues that must be taken into account to develop a case-based argumentation framework for open MAS.

This part of the document shows the conclusions that we have drawn from the state of the art revision, following this structure: Section 2.2 introduces important concepts of the argumentation theory that have been adapted to argue in AI systems; Section 2.3 makes a review of the argumentation study in the field of AI; Section 2.4 introduces current applications of argumentation in AI; Section 2.5 analyses the contributions of CBR to argue in MAS and summarises the conclusions of this analysis; Section 2.6 proposes open issues for the application of argumentation in agent societies and the uses of CBR to manage argumentation dialogues in open MAS and finally; Section 2.7 summarises the contributions of this chapter.

2.2 Argumentation Theory

Argumentation theory provides a framework to model such dialogical situations where a set of participants, which have opposing opinions about a certain claim, engage in a dialogue by generating arguments that support or attack this main claim. From its origins on the classic philosophy to nowadays, the argumentation study has given rise

to many argumentation theories. A theory for argumentation specifies the elements that define the argumentation dialogue; such as, for example, the argument components, the argumentation logic, the inference rules and the argumentation protocol. Currently, one of the most accepted argumentation theories is the *pragma-dialectical*, proposed by Van Eemeren and Grootendorst [van Eemeren and Grootendorst, 2004]. According to these authors:

'...Argumentation is a verbal, social, and rational activity aimed at convincing a reasonable critic of the acceptability of a standpoint by putting forward a constellation of propositions justifying or refuting the proposition expressed in the standpoint...'

Thus, *justifying* one's own opinion or *rebutting* others' is the main objective of argumentation. The argumentation process takes place over the proposition of a sequence of statements in favour of a claim (proarguments) or against it (counterarguments). Argumentation is aimed to increment (or decrement) the acceptability of a controversial point of view, in other words, to *persuade* the opponent (but in some situations collaborative decision making and negotiation are also possible objectives of the argumentation). Following van Eemeren and Grootendorst's point of view, the argumentation is part of a critical discussion. Its objective is to resolve opinion disagreements over a process that has the following stages:

- Confrontation: where the problem is presented.
- Opening: where the valid rules are determined (how to present evidences, which sources of facts are valid, how to evaluate differing opinions, termination rules, etc.).
- Argumentation: where the logical principles are applied in accordance with the rules that have been established and each party defends its position or attacks other positions.
- Concluding: where the parties reach an agreement (or do not) and the process ends.

Many MAS where the interaction between the agents is defined via an argumentation process (such as those reviewed in this chapter) have characteristics of the pragma-dialectical theory and, more or less explicitly, show regard for its stages in the dialogue

between the agents. In addition, several well-known concepts of the argumentation theory have been adopted for the AI community to manage the argumentation between the agents of a MAS. Among them, the dialogue typology of Walton and Krabbe [Walton and Krabbe, 1995], the theory of *dialogue games* and the theory of *argumentation schemes* are the most widely applied. Next sections provide a revision of these concepts.

2.2.1 Dialogue Typology

The philosophers Walton and Krabbe established in [Walton and Krabbe, 1995] a dialogue typology that distinguishes different types of dialogues regarding their initial situation, their main objective and the motivation of the participants. The main dialogue types identified in this work were: *persuasion*, *negotiation*, *information seeking and sharing*, *deliberation*, *inquiry* and *eristics*. Table 2.1 shows a summary of the Walton and Krabbe dialogue typology. Following, each type of dialogue is described:

- The *persuasion* dialogue is used when a participant tries to convince another of accepting certain proposition or statement. This could make the opponent to change its beliefs or to accept an action proposal.

- The *negotiation* dialogue takes place when several interested parts need to reach an agreement about sharing some scarce resource to be able to reach their objectives. Each part can have opposing interests. Therefore, despite trying to reach the agreement, the negotiating parts hope to achieve the most beneficial deal for their own interests.

- The *information seeking and sharing* dialogue occurs when one or more participants want to obtain an unknown information. In this dialogue it is assumed that each participant believes that other participants have no knowledge about the desired information. Therefore, participants do not take specific positions and collaborate to find the unknown information.

- The *deliberation* dialogue takes place when several participants try to reach a joint decision about an action or curse of action to perform. All participants are responsible to take the final decision and thus, they collaborate to reach to the best alternative.

- The *inquiry* dialogue is used when a participant wants to get certain unknown information, but, in this case, it is assumed that some participant of the dialogue knows this information (e.g. an expert).

- The *eristics* dialogue takes place when several participants meet each other trying to get the victory. In this case, the participants have conflicting positions and the objective of each one is to attack the others and to win the debate in the opinion of an audience.

Type	**Subtypes**	**Initial Situation**	**Main Objective**	**Objective of the Participants**
Persuasion	Dispute; Formal discussion; Proposal discussion	Conflicting points of view	Verbal resolution of conflicts	Persuade
Negotiation	Make deals; Global agreements	Conflict of interests; Need of collaboration	Negotiate	Take the maximum benefit
Information seeking and sharing	Scientific search; Research; Examination	General ignorance	Increase the knowledge and the agreement	Find or refute a proof
Delibera-tion	Purpose discussion; Board meetings	Need of action	Reach a decision	Influence the results
Inquiry	Expert consultation; Didactic dialogue; Interview; Interrogation	Personal ignorance	Extend the knowledge; Reveal positions	Win, share, show or extend the personal knowledge
Eristics	Eristic discussion; Dispute	Conflict; Antagonism	Reach a provisional agreement in a personal relation	Attack the opponent; Win in the opinion of an audience

Table 2.1: Walton and Krabbe Dialogue Typology[Walton and Krabbe, 1995].

From a more specific approach, to characterise a dialogue type we need to specify its main objective and the rules that guarantee the achievement of this objective. These elements form the *normative model* of the underlying dialogue type. Therefore, at this level, the dialogue types would be equivalent to the *dialogue games* that we present in the next section. The dialogue types shown in this section represent, from a more general point of view, the abstract types of the objectives and agreement rules that a participant should ideally meet when he or she is engaged in certain type of conversation and is willing to be reasonable and cooperative.

2.2.2 Dialogue Games

Dialogue games are interactions between two or more players, where each player 'moves' by making statements observing a pre-defined set of rules [McBurney and Parsons,

2002a]. They are a specific type of games of the *game theory* that are different from the classical games studied in the economy research area in the sense that the profits or losses for the victory or defeat are not considered. Another important difference is that in dialogue games participants are not able to model the potential moves or other participants by using some uncertainty measure, for instance, some probabilistic measure. Dialogue games have been used with multiple purposes in computational linguistics, AI [Bench-Capon, 1998] and philosophy (concretely in argumentation theory [Hamblin, 1970; MacKenzie, 1978]). In CBR systems dialogue games have been applied to model human reasoning about legal precedents [Prakken and Sartor, 1998]. In MAS, their more recent and successful application consist in 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, 2005b; Wardeh et al., 2008], *negotiation* [McBurney et al., 2003; Sadri et al., 2001a; Karunatillake et al., 2009], *inquiry* [McBurney and Parsons, 2001] and *deliberation* [McBurney et al., 2007].

A particular element of dialogue games, *commitment stores*, has been widely used in the area of MAS. The fact that an agent utters certain proposition during the dialogue means that this agent incurs 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 represent beliefs that do not necessary correspond with the actual beliefs of the participant. Furthermore, commitments may not hold out of the dialogue context.

Other 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 certain curse of action (they are *action commitments*). In this case, the commitments made during the dialogue can force the

participants to perform certain actions out of the dialogue context. For these authors, commitments can also represent the fact of uttering statement in the dialogue. Therefore, propositional commitments are viewed as a specific type of action commitments.

Finally, a different approach for commitments was presented by Singh in [Singh, 2000], who proposes a social semantics for the agent communication languages. According to Singh, the participants of the dialogue have to express their *social commitments*. These commitments represent their beliefs about certain propositions and their intentions to execute actions in the future.

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 to any type of dialogue does not exist yet. However, there are a common set of requirements among the models based on dialogue games which define their *syntax*. Based on Maudet's requirements and the approaches found in the literature, in [McBurney and Parsons, 2002a] a definition for the components that a dialogue game should have is proposed. However, a different view of the elements of dialogue games is presented in [Prakken and Sartor, 1998].

The components of the approach of McBurney and Parsons are the following:

- *Commencement rules*: rules that define the circumstances under which the dialogue commences.
- *Locutions*: rules which indicate what utterances are permitted. In legal contexts, for instance, locutions allow participants to express propositions, opponents to refute these propositions and again participants to refute this rebuttal justifying their propositions. Justifications imply to present proofs or arguments that defend these propositions.
- *Rules for combination of locutions*: rules that define the dialogical contexts under which particular locutions are permitted or not, or obligatory or not.
- *Commitment rules*: rules which define the circumstances under which participants incur dialogical commitment by their utterances. Also, these rules define how commitments are combined when utterances incurring conflicting commitments are made.
- *Rules for speaker order*: rules which define the order in which speakers can make utterances.

- *Termination rules*: rules that define the circumstances under which the dialogue ends.

Following Prakken's approach the common elements of dialogue systems are:

- A *topic language* $\mathcal{L}_t$, closed under classical negation.
- A *communication language* $\mathcal{L}_c$, where the set of *dialogues*, denoted by $\mathcal{M}^{\leq\infty}$, is the set of sequences from $\mathcal{L}_c$, and the set of *finite dialogues*, denoted by $\mathcal{M}^{<\infty}$, is the set of all finite sequences from $\mathcal{L}_c$.
- A *dialogue purpose*.
- A set $\mathcal{A}$ of *participants*, and a set $\mathcal{R}$ of *roles*, defined as disjoint subsets of $\mathcal{A}$. A participant a may or may not have a, possibly inconsistent, *belief base* $\Sigma_a \subseteq Pow(\mathcal{L}_t)$, which may or may not change during the dialogue. Furthermore, each participant has a, possibly empty set of commitments $\mathcal{C}_a \subseteq \mathcal{L}_t$, which usually changes during the dialogue.
- A *context* $\mathcal{K} \subseteq \mathcal{L}_\sqcup$, containing the knowledge that is presupposed and must be respected during the dialogue. The context is assumed consistent and remains the same throughout a dialogue.
- A *logic* L for $\mathcal{L}_t$, which may or may not be monotonic and which may or may not be argument-based.
- A set of *effect rules* $\mathcal{E}$ for $\mathcal{L}_c$, specifying for each utterance $\varphi \in \mathcal{L}_c$ its effects on the commitments of the participants.
- A *protocol* P for $\mathcal{L}_c$, specifying the legal moves at each stage of a dialogue. It is useful (although not strictly necessary) to explicitly distinguish elements of a protocol that regulate turntaking and termination.
- *Outcome rules* O, defining the outcome of the dialogue. For instance, in a negotiation the outcome is an allocation of resources, in a deliberation it is a decision on a course of action, and in persuasion dialogue it is a winner and a loser of the persuasion dialogue.

The approach of McBurney and Parsons is prospective (looking forward to model systems that do not exist yet). 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 more suitable for modelling the dialogue between a set of heterogeneous agents whose interactions will determine the dynamics and operation of the system. In addition, Prakken's approach assumes a presupposed knowledge about the domain, which 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 of falsity of utterances. The semantics of a dialogue game have the following functions [McBurney and Parsons, 2009, Chapter 13]:

- To provide a shared understanding to participants of the meaning of utterances, sequences of utterances and dialogues.
- To provide a shared understanding to designers of agent protocols of the meaning of utterances, sequences of utterances and dialogues.
- To provide a means of studying the properties of the protocol formally and with rigour.
- To provide a means of comparing protocols formally and with rigour.
- To provide a means of readily implementing protocols in production systems.
- To help ensure that implementation of agent communications in open MAS is undertaken uniformly.

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 exists before the locution can be uttered and the post-conditions which apply after its utterance. Axiomatic semantics can be *public* or *private*. In the former, the pre-conditions and post-conditions describe states or conditions of the dialogue that are publicly observable by all its participants whereas in the later some pre-conditions or post-conditions describe states or conditions of the dialogue that are only observable by some participants.

Other type of semantics is called *operational* semantics. This semantics views the dialogue game protocol as an abstract state machine and defines precisely the transitions between states. The transitions are triggered by the utterance of each locution.

A third type of semantics is the *denotational* semantics. In this case, 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 semantics. Finally, there are 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 giving that player moves such that executing them guarantees the player can win the game, no matter what moves are made by the antagonist).

In this PhD thesis, 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 accessible to the participants of the dialogue. Also, we endorse the view of Hamblin and define our notion of commitments as propositional commitments that agents incur during the dialogue, with no effect once the dialogue is terminated. Regarding the elements of the dialogue system, we follow the approach of McBurney and Parsons, since our framework is prospective and intended to model a system that does not exist yet, but which operation will be defined by the interaction of its agents.

2.2.3 Argumentation schemes

Argumentation schemes represent stereotyped patterns of common reasoning which instantiation provides an alleged justification for the conclusion drawn from the scheme. These schemes present an structure for the different common forms of argumentation (precedent-based argumentation, analogies, etc.). The arguments of argumentation schemes adopt the form of a set of general inference rules by which, given a set of premises, a conclusion can be inferred. In addition, they also have into account the non-monotonic nature of arguments. An implication of this is that arguments can be valid or invalid when new information is available.

Toulmin's scheme [Toulmin, 1958] was one of the first works that proposed argumentation schemes. The main contribution of this scheme was the inclusion of several elements that allow to describe the roles that the premises can play in a concrete argument. In this way, arguments win expressiveness. Toulmin's scheme is formed by the following elements:

- Affirmation: conclusion that must be shown.
- Data: premises that support the affirmation.
- Justification: expression that authorises the inference move from the data to the affirmation.
- Support: credentials that certify the statement expressed in the justification.
- Refutation: proposition that, in case of being true, would refute the affirmation.
- Qualifiers: words or phrases that express the power of the veracity of the argument for the affirmation.

The main criticism to the Toulmin's scheme is that it does not propose a clear way for the opponent to attack its elements neither distinguishes between the different types of attacks that it can receive (affirmation rebuttal, inference process rebuttal, etc.). Other foundational works that identified different types of argumentation schemes are those of Perelman [Perelman and Olbrechts-Tyteca, 1969] and van Eemeren [van Eemeren and Grootendorst, 1984; van Eemeren and Grootendorst, 1992].

Finally, the work of Walton [Walton, 1996], who presented a set of 25 different argumentation schemes, was an important contribution that solved the problems that the Toulmin's scheme arises. Each Walton's argumentation scheme has associated a set of critical questions that, if instantiated, represent potential attacks to the conclusion drawn from the scheme. Elements of Walton's schemes are:

- Premises: statements from which the conclusion of the scheme is inferred.
- Conclusion: affirmation drawn from the premises.
- Critical questions: different ways in which the conclusion can be attacked.

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 Walton's argumentation schemes makes them very suitable to reflect reasoning patterns that agents can follow to bring about conclusions and, what is more important, to devise ways of attack the conclusions drawn from other agents. Therefore, in the case-based argumentation framework proposed in this PhD work, we include Walton's like argumentation schemes as knowledge resources that our agents can use to generate arguments and attacks to arguments.

2.3 Argumentation in AI

From the beginning of AI research, the development of knowledge-bases, reasoning methods and cognitive architectures able to reply human reasoning has been a core area of interest. The work done in such area is typically known as *common sense reasoning* research. A reasoning method of this kind must include the following features:

- The ability to manage uncertain knowledge.
- The ability to reason with knowledge that is assumed to be true or false in the absence of any evidence that shows the opposite, which is called *default reasoning*.
- The ability to reason quickly over a wide range of domains.
- The ability to reason and take decisions in presence of incomplete knowledge and subsequently, to revise the beliefs and decisions that were taken when concrete knowledge is acquired, which is called *non-monotonic reasoning*.

Initially, argumentation theory was adopted in AI due to the inability of the classic propositional logic to reason and give explanations in presence of uncertain or imprecise information [Reiter, 1980]. The main problem with classical logic is its monotonic condition, which implies that the acquisition of new information cannot modify the conclusions that were inferred to that moment and thus, it is not applicable as common sense reasoning method. This problem already appeared in rule-based expert systems, where several rules could conflict or even be invalidated by the acquisition of new information. The process of drawing conclusions by using rules that can be defeated by new information is called *defeasible reasoning*. When defeasible rules are linked up to reach to a conclusion, the *proofs* that support such rules turn into *arguments*. The arguments can defeat each others, given rise to an argumentation process. To determine the winning arguments, they must be compared by establishing which beliefs are *justified*. Therefore, argumentation theory has been studied in AI to deal with the process of argument searching and more concretely [Bench-Capon and Dunne, 2007]:

- To define the argument components and their interaction.
- To identify the protocols and rules that manage the argumentation processes.
- To distinguish between valid and invalid arguments.

- To determine the conditions under which further discussion becomes redundant.

[Bench-Capon and Dunne, 2007] and [Rahwan and Simari, 2009] provide an extensive review of the argumentation research that has been done in AI throughout the history. According to the authors, the foundations of argumentation in AI lie in the studies done to extend non-classical logic to manage argumentation, the argumentation models that are based on dialogue processes and the diagrammatic treatments of the argument structure. We refer the reader to this important works for further details.

However, among the work that promoted the use of argumentation in AI, the research done in legal reasoning really stands out. In fact, the results obtained in the eighties from the work performed by the CBR researchers on applying this methodology of reasoning to legal domains fostered the study of argumentation in the AI community. Law is one of the most important application domains of the argumentation theory. In this domain, a case represents a conflict between two parties that argue about their opposing positions in court. Each party tries to persuade the other to obtain a verdict that was favourable to its lawsuit. The concepts that characterise legal cases cannot be universally defined by valid and sufficient conditions, but they are *open-textured*. Each case interpretation is arguable and hence, experts are in disagreement about what its verdict should be. An AI system suitable for working with legal cases must be able to store and manage uncertain and incomplete information.

CBR fits these conditions perfectly. Moreover, the CBR methodology has a high resemblance with the way in which people argue about their experiences. Therefore, CBR has been successfully applied in many legal reasoning systems that are based on precedents. This is the case of Anglo-American law, which follows a *common law* legal system whose judicial standard, *stare decisis*, orders that similar cases must be resolved with similar verdicts. In this domain, CBR is the most common reasoning mode. However, the judicial standard does not specify how to measure the similarity between cases. In fact, the similarity is not static, but it depends on each person's point of view and objectives. Rissland [Rissland et al., 2003] report this and other features that make the legal domain an excellent area to study the CBR typical research issues (indexing, retrieval, similarity measurement, etc.).

In Europe, the research on legal domains of the argumentation research community has historically followed a different approach, centred on the design of Rule-Based Systems (RBS) and the specification of logical models of legal reasoning. This trend is mainly due to the *civil law* legal system of most European countries, by which the legal code

is specified in the form of laws transcribed in an statute or constitution that is not based on precedent cases. Among the first contributions in this area are the works of Gordon [Gordon, 1987], who developed a hybrid system of knowledge representation for defeasible reasoning and Prakken [Prakken, 1993], who studied the concept of preference among conflicting arguments. Subsequently, a model of legal reasoning among precedents was proposed in [Prakken and Sartor, 1998]. Also, Bench-Capon did research into the defeasibility of legal arguments, the concept of explanation on expert legal systems, the building of arguments by means of dialogue games and the definition of ontologies in legal systems of information [Bench-Capon, 1989; Bench-Capon and Visser, 1997; Bench-Capon, 1998]. Other important European contribution to argumentation in legal domains was the Reason-Based Logic (RBL) theory about legal reasoning proposed by Hage and Verheij in [Hage and Verheij, 1995]. Finally, Dialaw [Lodder and Herczog, 1995], an interesting framework that models legal reasoning as a dialogical process, was presented by Lodder and Herczog.

In 1991, the seminal works of the CBR researchers Ashley, Branting, Rissland and Skalak were published. The ideas that were spread on their projects established the basis of what is known today as *interpretive CBR*. The first and probably the most important interpretive CBR system of that time was Ashley's HYPO system [Ashley, 1991]. HYPO generates legal arguments by citing previous cases (precedents) as justifications of the conclusions about who should win a dispute in the domain of the trade secrets American law. Concretely, the system generates *3-ply arguments* that show the best cases to cite for each party, their distinctions (the differences between the current and the previous case) and their counterexamples (previous cases that are similar to the current case and that were resolved in the other party's favour).

The main contributions of HYPO were the definition of several *dimensions*, in terms of which arguing in law is common, and the design of methods to compare and contrast cases on the basis of their applicable dimensions. In HYPO, the most similar cases (most *on-point* cases) are those that share a major number of dimensions. Therefore, the disputes in HYPO are represented by their dimensions, in view of the lack of a sound domain theory that prescribes rules for taking decisions over any legal problem. The dimensions show a set of facts that reinforce or weaken the argument of a plaintiff for a certain conclusion. However, legal experts do not reach an agreement about the relative importance of the dimensions. HYPO, thus, helps in the resolution of a new legal dispute by relating it with previous cases and by generating the whole argumentation process that might be followed. In addition to this, HYPO also offers

a list of hypothesis (or *hypos*) that modify the problem by reinforcing or weakening the position of each party. These hypos may be very useful for the plaintiff or the defendant when preparing themselves for dealing with the potential arguments that the other party might generate against his position during the trial.

The HYPO's success gave rise to the subsequent development of several systems that share its problem analysing style. The most direct descendant of HYPO and probably the most elaborated is Aleven's CATO system [Aleven and Ashley, 1997], an intelligent learning environment that teaches law students to build arguments from cases. In addition to the typical functionalities of argumentation systems, CATO also includes other abilities that legal experts have and that law students must learn: such as to organise a written argument by topics using multiple cases and to generate arguments about the similarity between cases. To perform that, CATO extended the argumentation model of HYPO with a *factor hierarchy*. This hierarchy is a representation of the domain normative knowledge about the meaning of the case factors that experts have. CATO's factors are a simplification of HYPO's dimensions and represent a set of stereotyped facts that, according to the experts, influence the resolution of a case. Thus, a case in CATO is a set of unary factors that are labelled as favourable to the plaintiff or to the defendant. Other important contribution of CATO was the definition of methods to generate and select arguments having the knowledge contained in the factor hierarchy into account.

Initially, CATO was deployed to work in the HYPO's domain, the trade secrets American law. However, CATO's factor-represented case-based argumentation model was used afterwards in other systems that operated in different legal domains. The system BankXX [Rissland et al., 1993], which generates arguments in the domain of American bankruptcy law by performing a *best-first* heuristic search in a high-interconnected network of legal knowledge, is an example. Moreover, the work done in HYPO and CATO systems also gave rise to several projects whose research objectives were centred in the development of some specific processing functionality of the legal information contained in the cases. Some examples are the systems: SPIRE [Daniels and Rissland, 1997], which mixes CBR with retrieval information techniques to extract passages from textual legal cases that could contain relevant information of the opinion of several courts about certain legal cases; SMILE [Brüninghaus and Ashley, 2001], which uses text classifiers to automatically decide the factors that are applicable to a specific legal case and; IBP [Brüninghaus and Ashley, 2003], which determines the underlaying issues of a case and, on the basis of them, predicts the verdict of case-based legal pro-

cesses. Recently, SMILE and IBP systems have been integrated in a new system that is able to reason directly with textual legal cases. This system analyses the cases to extract the relevant legal information, then predicts their verdicts and finally, shows such prediction with an explanation [Brüninghaus and Ashley, 2005].

Other kind of argumentation systems that have their roots in the eighties and early nineties are the hybrid CBR-RBR (*case-based and rule-based reasoning*) systems. One of the first real hybrid CBR-RBR system is another descendant of HYPO, the CABARET system [Rissland and Skalak, 1991], which produces legal arguments in the domain of the taxes American law. Before CABARET's development, 'hybrid' systems had a main reasoning mode and when it failed, the systems switched to the secondary mode of reasoning. However, CABARET has a domain-independent architecture that includes two reasoners (one case-based and other rule-based) that are managed by an agenda controller that uses heuristic rules to dynamically alternate the control over them. The system uses various knowledge sources: a database of legal cases, a database of rules and legal predicates and a set of domain-independent control rules, which determine the sequence of tasks to perform by using the information gathered by the controller. Finally, other important hybrid CBR-RBR argumentation system is GREBE [Branting, 1991], which was a pioneer system in using the justifications of legal cases to create new arguments. GREBE is a system for legal analysis that reasons with portions of precedent cases in the domain of the Texas workers compensation law. The system builds explanations for the classification of legal cases as instances of specific legal predicates by using a *back-chaining* technique that combines rules and portions of precedents.

2.4 Current Applications of Argumentation in AI

Nowadays, the argumentation research in AI is experiencing a new reactivation, mainly motivated by recent and interesting contributions developed in MAS. On one hand, the argumentation theory has been studied in MAS to manage the agent's practical reasoning. Practical reasoning is a well-known area in philosophy, but which historically has received less attention in AI than the theoretical reasoning. This type of reasoning analyses which specific action should be performed in a particular situation, instead of the theoretical reasoning objective of deciding the truthfulness of beliefs. Moreover, practical reasoning does not presuppose, as theoretical reasoning does, that the fact of reaching an objective is always adequate or profitable, but it must select the best objectives to perform and decide afterwards whether their realization is worthwhile. This

fits the reality of a MAS, where each individual agent has its own point of view and its particular objectives and interests. However, the theoretical reasoning about the state of the world and the effects of the potential actions to perform is also essential. Therefore, both types of reasoning must be considered in MAS. In [Rahwan and Amgoud, 2006], an argumentation-based approach for practical reasoning has been proposed. In this work, Dung's abstract argumentation framework [Dung, 1995] is instantiated to generate consistent desires and plans to achieve them. The works developed by Atkinson in her thesis and hers subsequent research are also other important contributions to the modelling of argumentation processes that allow the agents to reason about what is the best action to execute [Atkinson, 2005b].

In addition, the argumentation techniques have been applied to manage the agents' autonomous reasoning and the interaction between them [Rahwan, 2006]. In open MAS, the introduction of new information may give rise to new arguments that reinforce or weaken certain beliefs. Therefore, the argumentation techniques can be used as a way of revising the agents' beliefs in presence of incomplete or uncertain information. The work proposed in [Capobianco et al., 2005], for example, apply argumentation to keep the consistence of the agents' mental state in changing environments by using an appropriate representation of the environment and a mechanism that integrates the new information in the beliefs update process. Argumentation has also been applied in MAS as a selection means between conflicting desires [Amgoud, 2003] and objectives [Amgoud and Kaci, 2004], as a qualitative means of reasoning about the expected value of the realisation of certain actions [Fox and Parsons, 1998] and as generator of plans [Hulstijn and van der Torre, 2004; Simari et al., 2004].

Moreover, argumentation provides MAS with a framework that assures a rational communication. The dialogue typology of Walton and Krabbe [Walton and Krabbe, 1995] has been adopted in MAS to classify the different types of dialogues between the agents depending on the objective of the interaction. Other concepts of the argumentation theory (i.e. *dialogue games* [Hamblin, 1970; MacKenzie, 1978] and *argumentation schemes* [Walton, 1996; Walton et al., 2008] have also been applied to structure the dialogue between agents with different points of view according to the interaction rules that have been previously agreed. Recently, a wide range of approaches that formalise interaction protocols by using different dialogue games have been published [McBurney and Parsons, 2002a]. Some examples of dialogue game protocols about specific types of dialogues are: inquiry [Hulstijn, 2000], persuasion [Atkinson, 2005a], negotiation [Sadri et al., 2001b] and deliberation [McBurney et al., 2007]. As pointed out before, argu-

mentation schemes have several characteristics that make them very useful in defining the communication between agents. In the case of Walton's argumentation schemes, the *critical questions* are arguments that can be presented by an opponent to criticise the claim that the scheme poses, thus providing the argument with a clear structure that reduces the computational cost of generating and evaluating arguments. [Reed and Walton, 2005] proposes a formal framework to specify argumentation schemes for agent's communication by using the markup language *AML*, based on *XML*.

Among current research on argumentation in MAS, to study the effect of argumentation strategies in the interaction between agents is also a recent trend. Here, there are different approaches to the study of strategies in argumentation frameworks. On one hand, preliminary works studied the concept of strategy as developing heuristics for move selection in argumentation dialogues. A first contribution was provided in [Bench-Capon, 1998]. In this work the author defines a *Toulmin dialogue game machine* and proposes some heuristics for move selection. The acceptability of the arguments is computed by using some Toulmin-like rules. A similar work is the one presented in [Amgoud and Maudet, 2002], which proposes heuristics for move selection on the context of persuasion and negotiation dialogues. This research defends a three-level approach of strategy, inspired on naturally occurring dialogues between humans. The levels identified are:

- maintaining the focus of the dispute.
- building one's point of view or destroying the opponent's one.
- selecting the method to fulfil the objective set at levels 1 and 2.

While levels 1 and 2 refer to *strategy* (planning the line of argumentation), level 3 refers to *tactic* (the mean to reach the aims fixed at the strategical level). Then, the account for strategy proposed follows three steps to develop strategies:

- define some agent profile: agreeable (accept whenever possible), disagreeable (only accept when there is no reason not to), open-minded (only challenge when necessary), argumentative (challenge whenever possible), or elephant child (question whenever possible).
- choose to build or destroy.
- choose some appropriate argumentative content.

Opposite to the former work, this work computes argument acceptability by using a more general Dung's-like argumentation framework. In a subsequent work, the author studied the notion of strategy for selecting offers during a negotiation dialogue [Amgoud and Kaci, 2005], proposing different agent's profiles and different criteria for the notions of acceptability and satisfiability of offers. Also, in [Amgoud and Hameurlain, 2006] it is argued that there is no consensus on the definition of a strategy and on the parameters necessary for its definition. Consequently, there are no methodology and no formal models for strategies. This work defends that a strategy is a two steps decision process: i) to select the type of act to utter at a given step of a dialogue, and ii) to select the content which will accompany the act. Thus, an agent tries to choose among different alternatives the best option, which according to its beliefs, will satisfy at least its most important goals. There are two types of goals: *strategic goals*, which help an agent, on the basis of the strategic beliefs, to select the type of act to utter; and *functional goals*, which help an agent to select, on the basis of the basic beliefs, the content of a move. Then, the work proposes a formal model for defining strategies. The model takes as input the strategic and the functional goals together with the strategic and basic beliefs and returns the next move (act plus its content) to play. Then, the model assesses each alternative by constructing the set of supporting arguments for each one and evaluating their quality.

The agent profiles of [Amgoud and Maudet, 2002] were also considered in [Kakas et al., 2005] to develop different types of strategies. This work proposes an argument-based framework for representing communication theories of agents that can take into account the conformance to society protocols, private tactics of individual agents, strategies that reflect different types of personal attitudes (agents' profiles) and adaptability to the particular external circumstances at the time when the communication takes place. Although the authors do not provide a clear structure and definition for their notion of agent society, social relations between agents are captured in the form of preference rules that affect the tactic component of an agent and help it to decide the next move in a dialogue.

On the other hand, a different approach follows a game-theoretic view to the study of argumentation strategies. This is the case of the work proposed in [Roth and Rotolo, 2007], where the probability of a conclusion is calculated using a standard variant of defeasible logic, in combination with standard probability calculus. In this approach the exchange of arguments is analysed with game-theoretic tools, yielding a prescriptive account of the actual course of play. Other game-theoretic approach for the study or

argumentation strategies in negotiation dialogues was presented in [Rahwan and Larson, 2009]. This approach uses the paradigm of *Argumentation Mechanism Design (ArgMD)* for designing and analysing argument evaluation criteria among self-interested agents using game-theoretic techniques. Mechanism design (MD) is a sub-field of game theory concerned with determining the game rules that guarantee a desirable social outcome when each self-interested agent selects the best strategy for itself.

The approach analyses strategy-proofness under grounded semantics for a specific type of arguments, the so-called *focal arguments* (the arguments that agents are specially interested in being accepted). In a preliminary work [Rahwan and Larson, 2008], the authors restricted the analysis to the case where agents use a specific type of preference criteria, the *individual acceptability maximising preference* criteria. Following this criteria, every agent attempts to maximise the number of its arguments that are accepted. In further research [Rahwan et al., 2009] the ArgMD approach has been applied to more realistic situations in which each agent has a single focal argument it wishes to have accepted.

The authors demonstrate for both preference criteria that if each agent's type (characterised as the set of argument that an agent can bring up) corresponds to a conflict-free set of arguments which does not include (in)direct defeats, the *grounded direct argumentation mechanism* for this argumentation framework is strategy-proof.

Opposite to the heuristic-based approaches, the goal of this game-theoretic approach is to design rules that ensure, under precise conditions, that agents have no incentive to manipulate the outcome of the game by hiding arguments or lying (how to ensure the truth in an argumentation framework).

Along this section has been shown how the argumentation techniques have been successfully used to reach agreements that assure the coherence of the agents' mental state and to structure their interaction in disagreement situations. Parsons *et al.* [Parsons et al., 1998] proposed a seminal theoretical framework that unifies argumentation-based reasoning and communication for negotiation in MAS. More recently, Rahwan *et al.* [Rahwan et al., 2003] analyses this and other argumentation-based negotiation frameworks. A wide review of the current situation of the argumentation research in AI has also been published in the special issue on argumentation of the journal *Artificial Intelligence* [Bench-Capon and Dunne, 2007] and in the book [Rahwan and Simari, 2009]. Moreover, an effort to consolidate the work done in argumentation languages and protocols, argument visualisation and editing tools and, generally, in argumenta-

tion frameworks for MAS, was performed by the ASPIC project[1]. As a result, a new standard for argument interchange in MAS, the *Argument Interchange Format (AIF)*, has been proposed to serve as a convergence point for theoretical and practical work in this area [Willmott et al., 2006]. All these advances show how the study of argumentation in AI, and more concretely in MAS, is currently a research area that has a high activity and a growing interest.

Furthermore, as it was shown in Section 2.3, the work done in CBR argumentation systems in the eighties and nineties promoted the study of argumentation in AI. From then on, the successful application of CBR to reach agreements in case-based legal confrontations suggests the power of this methodology to manage this kind of disagreement situations. Nowadays, the argumentation research in CBR continues being very active [Rissland et al., 2006] and, in fact, some approaches that integrate CBR in MAS to help argumentation processes have already been proposed. However, the contributions are still scarce and many research in the area remains to be done. Our point of view is that CBR can be very useful to manage argumentation in open MAS and devise argumentation strategies. To demonstrate the foundations of this suggestion and to clarify the work that has already been done, next section reviews the uses of CBR for argumentation in MAS. Regarding dialogue strategies for argumentation, our objective in this PhD thesis is to contribute on the development of this area of research by providing case-based heuristic strategies and evaluating its performance in a real scenario. The reviewed works about strategies for argumentation dialogues are mainly theoretic and with this evaluation, we try to empirically prove the advantages of following heuristic argumentation strategies in MAS.

2.5 Research approaches in CBR-based argumentation for MAS

Argument management (generation, selection, evaluation etc. of the components of arguments and the management of the dialogue itself) is a key issue to deal with in argumentation-based dialogues in MAS. Our view, supported by the successful applications reported in the previous section, is that CBR is a suitable methodology to argue in two-party disagreement situations. Furthermore, if the use of CBR is extended to manage argumentation in multi-party dialogues (dialogues between a group of agents), argumentation can also be enhanced with the reasoning and learning capabilities that

[1]European Union's 6th Framework ASPIC Project (IST-002307), `http://www.argumentation.org`

CBR provides. These features could promote and distinguish CBR from other approaches for argument management. Note that in what follows, our research is focused on the applications of CBR in dialogical contexts in MAS. Therefore, we assume that there are a set of agents engaged in an argumentation in a multi-agent environment. The applications of CBR to manage agents' autonomous reasoning are out of the scope of the research performed in this PhD work.

To date, few research has cope with the use of CBR methodology to facilitate the argumentation between the agents of MAS. The current approaches are focused on managing two types of dialogues between agents: argumentation-based *negotiation* and collaborative *deliberation*. In addition, none of these approaches consider the social context of agents. Thus, this section analyses them in an attempt to show the promising advantages of using CBR to aid argumentation in open MAS. Despite the scarcity of applications in this area, the frameworks that have been proposed still introduce important features and interesting advances to analyse.

However, the language employed by the authors to present their approaches is very varied. Therefore, before dealing with the analysis, we introduce several dimensions that aim at establishing a common terminology for our research, some of them inspired by the characteristics used in [Rahwan et al., 2003] to compare several argumentation-based negotiation frameworks. These dimensions are organised and studied in three areas of analysis (context, argumentation model and discussion) and show how each framework deals with the argumentation related issues:

Context:

- Implementation domain: domain where the framework has been implemented and tested.
- Dialogue type: type of dialogue according to the classification proposed in [Walton and Krabbe, 1995].

Argumentation Model:

- CBR objective: what are the main purposes for applying the CBR methodology in the framework.
- Case-base contents: what kind of information is stored in the cases.
- Argument generation: method used to generate arguments in the framework.
- Argument selection: method used to select arguments in the framework.

- Argument evaluation: method used to evaluate arguments in the framework.
- Interaction protocol: method used to control the interactions between the agents of the framework.

Discussion:

- Argumentation style: theoretical framework that has inspired the argumentation style employed by the authors.
- Assumptions: main assumptions made by the authors in the design of the system that implements the framework proposed.

For clarity purposes, the analysis performed in this section is mainly centred on the CBR features that support the argumentation functionalities. The specific details about the design of each system have little interest for the review performed in this chapter and we refer the reader to the related bibliography for further information. Finally, note that not every dimension is mentioned when explaining each framework, since some of them remain unclear or unspecified by the authors.

2.5.1 The PERSUADER system

Context

In her thesis, Katia Sycara developed the PERSUADER system, which acts as a mediator in the implementation domain of labour management disputes between a company and its trade union [Sycara, 1987], [Sycara, 1989], [Sycara, 1990]. This was a seminal framework that integrated for the first time concepts of argumentation theory and CBR to create a negotiation model in a MAS. PERSUADER uses a *mediator* agent that manages the negotiations between two agents representing the company and the trade union. The mediator dialogues with the parts trying to reach an agreement, which is a contract that is accepted by both agents. A contract consists of a set of attributes (e.g. salaries, pensions and holidays) whose value must be decided. Opposite to many systems of its time, PERSUADER studied the argumentation in a non-cooperative domain, where each agent has its own objectives and tries to derive its maximum own benefit from the negotiation. The main objective of the mediator and hence, the objective of the dialogue in this framework, is to negotiate with both agents and persuade them to collaborate.

Argumentation Model

The agents' negotiation model consists of: (a) an iterative process that implies the identification of potential interactions (either by communicating with the other agents or by reasoning about the current state of the negotiation and the agents' intentions) and (b) the modification of the own intentions to avoid the interactions that could hinder the agreement process. Thus, the model favours the cooperation. In addition, the mediator agent is able: (a) to represent and keep models of the agents' beliefs and preferences, (b) to reason with these beliefs and (c) to modify them to influence their behaviour. Therefore, in order to perform a *persuasive negotiation*, the mediator agent keeps a model of the company and trade union agents and by reasoning over it, tries to find all possible ways of affecting the behaviour of these agents.

In PERSUADER, one of the CBR objectives is to infer the model of beliefs and preferences of an unknown agent. In this way, the mediator retrieves the information about past negotiations with similar agents that was stored in precedent cases and adapts it to fit to the current context. Moreover, during the negotiation, the mediator agent can update the agents' models by observing their reactions to the arguments that it offers to them. To manage the negotiation process, the mediator agent has available two types of knowledge:

- Domain knowledge stored in the case-base: the contents of the case-base consist of negotiation concepts represented by attributes that are hierarchically organised in networks (negotiation issues, concrete negotiators, negotiators' objectives, negotiation context and final agreement).

- Reasoning knowledge: is the knowledge needed to evaluate the fairness of a contract or to improve it. This knowledge is represented in terms of multi-attribute utilities that are associated with the objectives of each agent and express its preferences and criteria for selecting among proposals.

The argument generation is performed by integrating several techniques: the search in an objective graph, the use of the multi-attribute utilities and the use of the case-base of precedent negotiations. In PERSUADER, persuading an agent to change its assessment about the proposal of certain contract entails producing an argument that increases the benefits that this agent receives with that contract. The received benefits can be inferred as a linear combination of the utilities that will be received for the

value of each attribute that composes the contract. This also determines whether the objectives of the agent have been fulfilled or not. Therefore, the global benefit that will receive an agent can be increased (and hence, the agent can be persuaded) by following two types of persuasion strategies:

1. Strategies to change the importance of a concrete objective:
    - To indicate a change (increment or decrement) in the contribution of the objective to fulfil an objective of highest level.
    - To indicate a change in the viability or efficiency of the objective.
2. Strategies to change the utility value of a concrete objective:
    - To retrieve from the case-base a counterexample that shows an opposite behaviour of the agent in a similar past negotiation.
    - To retrieve from the case-base examples of similar agents that accepted the proposed value of the objective in similar past negotiations.

Therefore, another CBR objective in PERSUADER is to retrieve past cases that act as arguments for persuading an agent to accept a specific contract. In case that one part tried to prematurely withdraw from the negotiation without having reached an agreement, in addition to these *appealing* arguments, the system is able to generate *threats*.

The negotiation process in PERSUADER starts with the introduction of the set of objectives of the company and the trade union that conflicts and the factors that define the negotiation context. Initially, the mediator agent generates a contract proposal and shows it to the agents. If the proposal is accepted, the negotiation ends. Otherwise, the mediator chooses either to generate another contract proposal (if none of the parts accept it) or to start a persuasive argumentation trying to persuade the agent that does not accept the proposal (if it is accepted by one of the agents). The output of the negotiation is the agreement of a contract or an indication that the negotiation failed after certain number of proposals.

Discussion

The PERSUADER argumentation style is based on the persuasion psychology of Karlins and Abelson [Karlins and Abelson, 1970] and, as it is pointed out before, there are

two types of possible arguments: appealings or threats. The main contribution of the system was the creation of the first model of persuasive argumentation between several, by default, non-cooperative agents with different interests that integrates elements of CBR, utility theory and argumentation theory. However, this was a foundational work where the agent's model was somehow specified by abstractions. Although the author justified such design decision by saying that it reduces the overload of building and maintaining real agent's models, the multi-agent condition of the system could be debatable. Anyway, the system is completely designed for two party mediated argumentation, which is inapplicable to open MAS, where the number of participants in the argumentation dialogue can change and the presence of a mediator cannot be assumed by default. Moreover, PERSUADER is highly domain-dependent and user-oriented, with reasoning algorithms based on the knowledge introduced on the system, argument selection completely determined by a pre-established hierarchy of argument types and user-based evaluation of arguments.

2.5.2 CBR for Argumentation with Multiple Points of View

Context

Nikos Karacapilidis et al. developed a model that integrates CBR and argumentation for supporting decision making in discussion processes. This model was implemented in the *Argument Builder Tool (ABT)* of the multi-agent framework for collaborative deliberation HERMES [Karacapilidis and Papadias, 2001], [Karacapilidis et al., 1997]. This is an *Argumentation-based Decision Support System (ADSS)* that helps a group of users (human agents) to build sound arguments to defend their positions in favour or against other alternative positions in a discussion. HERMES maps the argument process into a *discussion graph* with tree structure and shows graphically the possible discourse acts that the agents could instance. The system uses CBR to make the appropriate queries to the (internal or external) databases that store information that support the positions of the agents that participate in the argument and, thus, to generate discourse acts that successfully show their interests and intentions. However, as it is only a support system, afterwards the agents are free to adopt or not the ABT's proposals.

Argumentation Model

In this framework is the system itself who manages the interaction between the agents, being the CBR engine a reasoning component integrated in it. Therefore, the case-base is common for all agents and belongs to the system. The cases are flexible entities that store a set of argumentation elements that can be interpreted depending on the state of the discourse and each agent's point of view. The argumentation elements that HERMES considers are the following:

- Issues: decisions to take or objectives to fulfil. They consist of a set of alternatives.
- Alternatives: potential choices.
- Positions: predicates that either advocate for the choice of a concrete alternative or deviate the agents' interest for it. They can also refer to other positions and give additional information about them.
- Constraints: which represent preference relations over arguments *(< position, preference relation, position >)*, where the preference relation can be *more (or less) important than* or *of equal importance to* certain alternative.

The arguments in HERMES are either tuples < *position, link, position* > or < *position, link, alternative* >, where a positive link denotes an argument favouring an alternative and a negative link denotes a counterargument to it. The system evaluates them by using constraints that the users of the system introduce, checking previously the consistence of the new constraints in relation with the constraints that have been introduced before. This checking gives rise to new argumentation processes to solve the possible conflicts between constraints. The constraints establish a partial ordering over the positions and, in this way, the position with a highest weight (highest preference) becomes the winning argument.

The argumentation process in HERMES is performed by means of several discourse acts with different functions and roles. There are two types of discourse acts: (1) *agents' acts*, which represent the user's actions and correspond to the functions that the user interface of HERMES supports (e.g. opening of new issues, alternatives submission, etc.) and, (2) *system's internal acts*, which are consequence of the agents' acts and represent functions for discourse consistency checking, discussion update and solution recommendation. The latter functionality is performed by using CBR.

Therefore, the main objective of the CBR methodology in the system is to examine the current discussion and to suggest the participants the best discourse acts to fire, according with their points of view and preferences. Thus, the contents of the HERMES case-base represent past argumentation processes. The cases consist of the following elements:

- Situation: relevant part of the discussion at the time when the case was stored in the case-base. It characterises the concrete problem that the case represents and consists of the elements:
  - Target: element to be argued about by the agent (issue, alternative or position).
  - Discussion: discussion that the case has been extracted from (represented by a link to the relevant part of the discussion tree that the case refers to).
- Solution: alternative or position that the system proposes to argue about the target.
- Evaluation: suitability of the case to the agent's agenda.

The case-based argumentation process in HERMES consists of the following phases:

1. Intention submission: first, agents submit their intentions by declaring the arguments that they want to include in the argumentation and by mapping their point of view about the current discussion. Therefore, the arguments are manually generated by instantiating some of the pre-established discourse acts that the system has.
2. Cases retrieval and selection: then, HERMES retrieves those cases for which the target coincides with the agents' current argumentation objective. Afterwards, the system performs a case selection based on the agents' point of view and the current state of the argumentation.
3. Cases adaption: this is a semi-automatic process, being the user who selects the cases that need adaption among the set of similar cases that the system has proposed.
4. Argument assertion: finally, by using the retrieved and adapted cases, the agents provide warrants to their assertions. Argument assertion involves firing the ap-

propriate discourse acts to propagate the information in the discussion graph and to retain the new case.

Discussion

The argumentation style of HERMES is inspired by the ZENO's model of argumentation [Gordon and Karacapilidis, 1997], which is based on the IBIS informal-logic argumentation framework [Rittel and Webber, 1973]. Therefore, HERMES interprets arguments on the basis of their intentional context by using an informal logic. The authors view the system as a MAS because it can be used by several human agents that interact between them and with the system by means of the HERMES user interfaces. An important contribution of HERMES was the proposition of this new case-based argumentation support functionality, which, in view of the discussion current state, produces the best arguments to support the assertions of the participants taking their points of view and preferences into account. Moreover, the user interface allows to follow easily the course of the argumentation. However, the framework does not define a specific interaction protocol between the agents. In addition, the arguments evaluation depend on constraints that determine a preference relation among them, also introduced by the users. Therefore, the good-end of the argumentation process entirely depends on the honesty, experience and disposition to collaborate of the users, which cannot be assured in open environments. Human intervention is compulsory almost at any time. These assumptions pose heavy difficulties to adapt this framework to work in open MAS with a changing number of heterogeneous software agents participating in the dialogue.

2.5.3 Case-based Negotiation Model for Reflective Agents

Context

Leen-Kiat Soh and Costas Tsatsoulis designed a case-based negotiation model for reflective agents (agents aware of their temporal and situational context). This model uses CBR to plan/re-plan the negotiation strategy that allows the most effective negotiation on the basis of past negotiations [Soh and Tsatsoulis, 2001a], [Soh and Tsatsoulis, 2001b], [Soh and Tsatsoulis, 2005]. In this framework, a set of situated agents that control certain sensors try to track several mobile targets. The aim of the agents is to

coordinate their activities and collaborate to track the path to as many targets as possible. The agents' sensors have limited power and coverage and each agent only controls a subset of sensors. Although the cooperativeness is assumed, each agent has individual tasks to fulfil. Therefore, when an agent has not enough coverage or power capabilities to track a target, it needs to negotiate and persuade other agents and achieve that they leave their tasks and help it to track the target. Hence, the framework was implemented to solve a typical problem of limited resources allocation.

Argumentation Model

The agents of this model are autonomous entities that own two separated and private case-bases. Each agent has a *CBR manager* that performs 3 functions: (1) retrieves past negotiation cases that are similar to the current negotiation, (2) uses them to determine new negotiation strategies and (3) maintains the case-base. The CBR manager allows agents to learn to negotiate more effectively by using the knowledge of past negotiations. The cases contents store descriptions that characterise the agents' context in a previous negotiation. Concretely, a case is composed by the following elements:

- Description of the part of the world relevant to the case: information about the sensors and the target.
- Agent's profile: sensor power, activated sensor sector, state of the communication channels, task list, etc.
- Neighbours' profile: agent's knowledge about its neighbours.
- Negotiation outcome.

An agent can play two possible roles: as *initiator* or as *responder* of the negotiation. As initiator, the agent tries to negotiate with its neighbours to reach its targets. As responder, the agent receives a negotiation proposal and answers it. Therefore, the information gathered by the agent when playing each role is stored in the corresponding case-base. When an agent determines that it cannot reach a target by its own, it tries to collaborate with other agents initiating an argumentation dialogue. In this framework, the arguments are pieces of information that an agent sends to its neighbours to persuade them to share certain resource with it. Persuasion implies to surpass the *negotiation threshold* of an agent and convince it to share some of its resources.

In order to achieve it, the initiator agent retrieves cases that represent negotiations that are similar to the current one from its *initiator case-base*. The cases are retrieved by comparing their *case descriptors* with the characteristics of the new negotiation. The *negotiation strategy* that will follow the agent, which is deduced from the *negotiation parameters* of the cases, specifies which type of information should be sent to convince quickly a particular responder agent. Initially, the initiator agent retrieves several strategies and afterwards, it uses *Multivalued Utility Theory*. By using this theory agents are able to relate constraint satisfaction criteria of the negotiation at hand with the potential negotiation strategies and to select the one that optimises such criteria while minimising the risky behaviour.

In this model the cases are situated, thus, the CBR manager must adapt the negotiation parameters to fit to the current negotiation. To that purpose, it uses certain *domain-dependent adaption rules* that the framework pre-establishes and, once the adaption is performed, the negotiation starts. The arguments to send to other agents are ranked and selected on the basis of certain *selection rules* also pre-established in the framework. Following them, the initiator agent selects which piece of information (arguments about the world, the target or the agent itself) can be most effective to persuade a specific agent in the current context. In each negotiation step, the initiator agent sends an argument to the responder agent. In its turn, the responder agent evaluates the evidence of each argument by using the information of its *responder case-base* and a *relaxed constraint satisfaction* approach. The process continues until a maximum number of steps is reached or an argument surpasses the *negotiation threshold* of the responder agent and it is accepted. The interaction protocol is defined by means of a series of states over which the negotiation takes place. At any time, the negotiation parameters that the CBR manager adapts have a high influence in the sequence of states that the negotiation follows.

Discussion

The framework of Soh and Tsatsoulis does not interpret persuasion as most legal argumentation systems do [Jakobovits and Vermeir, 1999]. These perform a detailed study about the dialectical context of legal arguments and their condition of proposals, defenses, positions or attacks. However, the argumentation style of this framework views persuasion as a negotiation protocol of information interchange between two agents that try to reach an agreement by using an argumentation process. To describe the logical

framework, the multicontext BDI framework of [Parsons et al., 1998] was extended. In addition, temporal characteristics were included by using a temporal logic that defines relations over time intervals. Other important contribution of this framework was the introduction of learning capabilities for the agents by using the CBR methodology. Moreover, the negotiation strategy is inferred dynamically from the information of the case-base and concurrent negotiations are allowed.

However, the model assumes certain characteristics that can pose several drawbacks to its application to open MAS. On one hand, the neighbours and their controlled sensors must be known in advance. On the other hand, despite concurrency is admitted, the agents can only negotiate about one issue at the same time. Finally, the framework has strong assumptions about the honesty, cooperativeness and rationality of the agents that do not fit the reality of open MAS.

2.5.4 Argument-based selection Model (ProCLAIM)

Context

Pancho Tolchinsky et al. extended the architecture of the decision support MAS for the organ donation process CARREL+ [Vázquez-Salceda et al., 2003] with ProCLAIM, a new selection model based on argumentation [Tolchinsky et al., 2006a], [Tolchinsky et al., 2006c], [Tolchinsky et al., 2006b]. In CARREL+, a *donor agent (DA)* and a set of *recipient agents (RAs)* argue about the viability of the organ transplant to some recipient. If an agreement is not reached, the organ is discarded. ProCLAIM includes a *mediator agent (MA)* that controls the collaborative deliberation dialogue and uses a *CBR engine* to evaluate the arguments about organ viability that the agents submit. The final decision must fulfil several guidelines that, in ProCLAIM case, are the human organs acceptability criteria that CARREL stores in the *Acceptability Criteria Knowledge Base (ACKB)*.

Argumentation Model

The mediator agent uses a case-base to store all relevant information about past donation processes. The cases contents consist of a *description* of the transplant medical information and an *argument graph* that shows the arguments that were submitted by the agents that participated in the donation process. Different cases can share the same

argument graph. In addition, an argument graph has an *evidential support* $< F, K >$, where F stands for the certainty on the correctness of the final decision that was made and K for the number of cases that share the graph. This certainty corresponds to the transplant phase when the decision was made; phase 1 includes potential arguments that were submitted before the organ extraction, phase 2 includes conclusive arguments that were submitted after the organ extraction and phase 3 includes conclusive arguments about the final result after the organ transplant.

The MA tasks consist of (1) directing the possible dialectical movements of each agent, (2) ensuring that the submitted arguments observe the guidelines and (3) using CBR to assign intensities to the arguments, offer new relevant arguments and take a final decision about the winning arguments. ProCLAIM arguments are instantiations of the *argument schemes* and *critical questions* of the CARREL+ knowledge resource *Argument Scheme Repository (ASR)*. This repository completely characterises the space of possible arguments that the agents can submit. Moreover, agents use a 1st order logic programming language to generate arguments [Modgil et al., 2005]. When a new organ is offered, the DA submits to the MA its arguments about the organ viability. Afterwards, the RAs counterargue and the MA must take a final decision. To that moment, the arguments are of phase 1. Once the organ is extracted, a RA can change its mind and consider it as non-viable. Therefore, the RA must submit more phase 2 arguments to support its position. Finally, if the organ is transplanted and complications arise, the RA submits more phase 3 arguments to explain the failure or the actions performed to solve them and achieve a successful transplant.

When the DA and the RAs have submitted their arguments for the organ transplant, the MA evaluates them. If there are conflicts, the MA compares the resulting argument graph with those stored in its case-base to retrieve similar graphs and decide the winning arguments. Once the similar argument graphs are obtained, the MA checks the descriptors of the cases that the graphs have associated and rules out the ones that contain arguments that do not observe the guidelines. In addition, the graphs whose evidential support falls below certain threshold are also rejected. In some exceptional cases, the CBR engine could find valid arguments that were turned down by the guidelines. Even arguments that were rejected by both the CBR engine and the guidelines can be accepted because the agent who submitted them has either a high reputation, a specific role that increases the confidence on its opinion or a certificate that supports its arguments. This knowledge is encoded in the *Argument Source Manager (ASM)* resource and can be acceded by the MA to readjust the strengths of the arguments.

The potential conflicts between the evaluation of arguments proposed by the different CARREL+ knowledge resources are solved with a pre-established *resource preference relation*. Finally, the resulting set of argument graphs is put together in a *solution graph* that represents the CBR engine decision concerning the viability of the transplant to a specific recipient agent. If this argument graph was already included in the mediator's case-base, the description of the new case is associated with it and, hence, its evidential support increases. Otherwise, the new graph and its associated case are added to the case-base.

Discussion

ProCLAIM argumentation style is based on Dung's abstract argumentation framework. An interesting contribution of this model is that it can increase the organ acceptance rate by allowing the donor and the recipients to argue about their decisions. In addition, it also allows exceptions over the human organ transplant acceptability guidelines. The ProCLAIM model has also been recently applied to wastewater management [Aulinas et al., 2007] (also a collaborative decision making domain with critical features) and to deliberate over action proposals [Tolchinsky et al., 2007]. However, in all applications the argumentation process depends on the contents of the knowledge resources that the MA accedes. Therefore, an intensive effort to acquire and maintain such knowledge must be performed. Moreover, as pointed out before, the space of possible arguments to submit is completely characterised by the ASR. This implies that the agents have limited expressiveness, although the authors state that this decision has been taken for security reasons in the critical domains where the model operates. As Sycara's PERSUADER system, this is also a framework that was designed for performing a mediated argumentation, but opposite to PERSUADER, the cooperative nature of the agents is assumed by default. Therefore, this heavy assumption hinders again the adaption of this system to open MAS.

2.5.5 Argumentation-based Multi-Agent Learning (AMAL)

Context

Santiago Ontañón and Enric Plaza developed the *Argumentation Based Multi-Agent Learning (AMAL)* framework [Ontañón and Plaza, 2006], [Ontañón and Plaza, 2007].

The agents of this framework are autonomous entities able to independently solve classification problems and to learn by experience, storing the knowledge acquired during the solving process in their private case-bases. The set of possible classification classes is predefined in the framework. The aim of the interaction between the agents is to increase the solution quality by aggregating the knowledge of a group of expert agents. Therefore, they engage in a collaborative deliberation dialogue.

Argumentation Model

When an agent receives a new problem to solve, it firstly uses the CBR methodology to retrieve the most similar cases from its case-base and provide an initial solution to the problem. In this way, the problem is classified into the class that the most similar cases belong. In this framework, the cases contents consist of a set of attributes that describe a problem and the solution class that classifies it ($C =< P, S >$). Moreover, the AMAL agents are able to provide a *justified prediction* ($J =< A, P, S, D >$) that explains the reason why a certain solution has been proposed to solve a specific problem. An agent A generates the justified prediction J to assert its belief that S is the correct solution for the problem P and D is the *evidence support* for such statement. This evidence is a symbolic description that contains the relevant information (common attributes) shared by the problem and the retrieved cases whose class is S. Therefore, D stands for the claim that all or most of the cases that are similar to P in the agent's case-base belong to the class S. The justified predictions can be generated by using CBR in combination with any other learning method, as decision trees or *Lazy Induction of Descriptions, LID* [Armengol and Plaza, 2001], the latter being the method used in AMAL.

Both cases and justified predictions are information pieces that AMAL agents use to generate three types of arguments:

- Justified Predictions, as previously explained.
- Counterarguments, which show the evidence that other agent has for classifying the problem as belonging to a different class.
- Counterexamples, which are cases sent by other agent to contradict an argument by showing an example case that classifies the problem into a different class.

By means of the arguments, agents get involved in a global solving process that increases

the quality of the initial solutions proposed by each agent by reaching an agreement about the correct solution for the problem. To achieve this, a *preference relation* to evaluate the arguments that bring into conflict is needed. In AMAL, this preference relation is based on a global *confidence measure* that is computed for each justified prediction. In the face of several conflicting arguments, the winning argument is that with higher global confidence (the one that classifies the problem into the class that has been predicted by the major number of agents).

The deliberation process in AMAL takes place across an interaction protocol that defines a series of rounds. In each one a *token passing* mechanism is used to specify which agent is authorised to interact with the others. During its round, an agent can either assert an argument or rebut it with a counterargument (or counterexample). Agents can also assert arguments when they accept an incoming argument and change their prediction. The AMAL protocol allows agents to include the counterexamples that receive in their case-bases and, hence, to increase their knowledge. The interaction ends when an agreement is reached or when a maximum number of rounds is surpassed. In the latter case, the final solution for the problem is decided by using a weighted voting mechanism.

Discussion

The AMAL framework is a newly contribution to the study of argumentation-based learning models for MAS whose agents have individual learning capabilities. This model also differs from many other argumentation frameworks on its dynamic computation of the relation preference between arguments. In addition, the argumentation style is completely case-based. However, the framework assumes honesty, cooperativeness and rationality in the agents' interactions, which cannot be assured in open MAS. All agents must have at least some knowledge about the problem to solve. Therefore, this framework is not conceived for open distributed environments, where the heterogeneity of agents makes non-assumable that all participants in a dialogue have a minimum knowledge about the problem at hand. Obviously, in order to take the maximum profit from this approach of learning from communication, the knowledge must be conveniently distributed over the agents' case-base. If all agents become experts, collaboration and learning from others would be a nonsense.

2.5.6 General Analysis

The main properties of the frameworks that this chapter has reviewed are summarised in the Table 2.2 (referring the dimensions previously presented and analysed in this section). The diversity of domains and purposes in applying both argumentation and the CBR methodology on each framework makes difficult to perform a formal comparison that expresses which is better at solving a particular type of problems. Therefore, the table aims at summarising and clarifying how each framework deals with the argumentation related issues. Nevertheless, some similarities and differences between them can still be identified.

As pointed out before, the implementation domain differs almost on each framework, being HERMES and ProCLAIM the ones that somehow share a common purpose: to provide decision support for a group decision-making. In addition, among other applications, both have been implemented and tested in the medical domain [Karacapilidis and Papadias, 2001], [Tolchinsky et al., 2006b]. In this dimension, the main difference between them is that HERMES helps agents to select the best argument to instantiate in a particular context and hence, to win the discussion, while in ProCLAIM the system assists the mediator agent (and not the donor agents) to decide which agent has posed the best argument and should be the winner of the discussion. Therefore, although working in a similar domain, these systems are aimed at solving different subproblems inside the more general problem of supporting group decision-making.

Similarly, although HERMES, ProCLAIM and also the AMAL framework share the same dialogue type (deliberation), the final objective of the interaction between the agents of these systems is quite different: HERMES is mainly centred on the argument diagramming and its graphical representation, helping agents to follow the discussion and supporting them with tools to pose better arguments; ProCLAIM deals with the internal deliberation of the mediator agent, supporting only this agent to make the best decision among the set of potential winners and finally; in the AMAL framework all agents have the common objective of deciding the best classification tag for a specific object and act as a group of experts that cooperate by aggregating their knowledge in the deliberation process. In the same way, PERSUADER and Soh's frameworks also share the dialogue type (negotiation), but from a different perspective. Thus, while in PERSUADER the mediator agent completely centralises the negotiation process and the company and the trade union do not keep a direct interaction, in Soh's framework all agents are autonomous and able to play an initiator role that starts and manages a

Framework / Dimensions	PERSUADER	HERMES	SOH's	PRoCLAIM	AMAL
Implementation domain	Labour conflicts mediation	Decision support (e.g. Deciding treatments for patients and Planning cyclepaths)	Dynamic resource assignment in a sensorised environment with mobile targets	Decision support (e.g. Organ transplant and Wastewater management)	Classification problems
Dialogue type	Negotiation & Persuasion	Deliberation	Negotiation	Deliberation	Deliberation
CBR objective	Provide information for building agent's models. Generate precedent-based arguments	Provide argument warrants. Assist in argument generation	Select negotiation strategies	Argument evaluation. Offer other relevant arguments	Individually solve the problem. Select the global solution. Learn by experience
Case-base contents	Concepts of previous negotiations hierarchically organised in networks	Set of attributes representing past argumentation processes	Descriptions of the agents' context in a previous negotiation	Medical information of past transplants and argument graph of their donation processes	Description of previous problems and their solution class
Argument generation	Case-based, objective graphs search and multiattribute utilities	Manual generation	Multivalued attribute utility theory	Instantiation of the argument schemes and critical questions of Walton (1996). 1st order logic programming language (Modgil *et al.*, 2005)	Lazy Induction of Descriptions (Armengol & Plaza, 2001)
Argument selection	Pre-established conviction power hierarchy	Helped by the ABT proposals	Domain-dependent selection rules	Preference relation among knowledge resources	Selection implicit in CBR cycle
Argument evaluation	User-based evaluation	User-based evaluation	Constraint relaxed satisfaction. Context-dependent rules	Guidelines-based and precedent-based evaluation performed by the MA	Preference relation based on a case-based dynamic confidence measure
Interaction protocol	Non-specified. Defined by the mediator's decisions	Non-specified. Defined by the discourse acts	Defined by the state diagram. Influenced by the CBR manager decisions	Guided by the MA. Dialogical movements determined by the ASR argument schemes instantiation	Argumentation-based Multi-Agent Learning Protocol
Argumentation style	Karlins persuasion psychology (Karlins & Abelson, 1970)	Zeno's informal logic (Gordon & Karacapilidis, 1997)	Persuasion. Parson's et al. logic framework (Parsons *et al.*, 1998)	Dung's argumentation framework (Dung, 1995)	Case-based multi-agent learning
Assumptions	Human agents. Unique mediator's case-base	Unique system case-base. Human agents. CBR is a system component	Shared ontology, Agents are Homogeneous, cooperative, autonomous, reflective, honest and rational. 2 private case-bases per agent	Agents are collaborative. Control centralised by the MA. Unique MA's case-base	Shared ontology, cooperative and autonomous agents, shared objective. One private case-base per agent

Table 2.2: Main features of the analysed CBR-based argumentation frameworks.

direct dialogue with other agents.

With respect to the CBR objective, in all frameworks the CBR methodology has been mostly used to generate, select or evaluate arguments on the face of previous similar experiences. Consequently, as in any CBR system, the contents of the case-base in each framework consist of a set of elements that describe these previous experiences. However, the power of case-based learning can be better well-spent and could be used, for example, to perform a strategic argumentation. PERSUADER and Soh's frameworks have proposed the first preliminary attempts to develop case-based argumentation strategies by learning agent's profiles and hence, easily persuading specific types of agents. In these frameworks, the contents of the agent's case-base store some features about other agents that were observed during previous negotiations. These features are used afterwards to enhance similar negotiations by having agents' beliefs and preferences into account (e.g. PERSUADER) or by sending the most useful information to persuade a specific agent (e.g. SOH's framework). However, in both frameworks the application domain has a decisive influence on the argumentation strategy. Thus, in PERSUADER arguments are presented following a domain dependent argument hierarchy while in Soh's framework cases are situated and need domain-specific adaption rules to devise strategies that are applicable to the current negotiation context from them. As will be shown in the next section, argumentation strategies have to be further elaborated and future work remains to be done in this area.

Moreover, the application domain of each concrete framework and the contents of the case-base have a decisive influence in the entire argument management process (case-based generation, selection and evaluation of arguments). Only the AMAL framework performs a completely case-based argument management. Note that the agents of this framework use their case-base at every step of the argument management process (although they generate arguments by using the LID technique [Armengol and Plaza, 2001], this method makes use of the information stored in the cases). The other frameworks include different techniques to manage the generation, selection or evaluation of arguments (or the user manually performs them). Furthermore, in PERSUADER, HERMES and ProCLAIM, all responsibility about the management of the argumentation process falls on one agent or on the system itself, while the rest of participants in the dialogue are human entities that are only in charge of posing arguments in favour of their objectives. It is not demonstrated that the methods and argumentation theories that are valid for a single agent were also valid if the system was entirely automated by creating real autonomous software that act on humans behalf. On most cases, the

argumentation that is conducted by a mediator totally depends on its capabilities and features. Therefore, multi-agent argumentation and moderated argumentation could need from different argumentation theories and methods with every likelihood. The differences between both types of argumentation can be even greater when working with open MAS.

In the argumentation style dimension, all approaches differ. On one hand, although PERSUADER and Soh's frameworks are based on the theory of persuasion, the former follows a formal psychological theory proposed by Karlins and Abelson [Karlins and Abelson, 1970], which studies the different types of arguments that are thought to be persuasive in negotiation processes between humans, while the latter views persuasion as a negotiation protocol to coordinate the interaction between two agents that want to reach an agreement in a resource allocation problem. This protocol is based on the formal argumentation framework for negotiation and reasoning proposed by Parsons [Parsons et al., 1998]. On the other hand, the frameworks intended for deliberation also show important differences in the foundations of their argumentation style: HERMES is based on Zeno's informal logic [Gordon and Karacapilidis, 1997], which proposes an argument-based labelling function to evaluate the quality of the potential positions proposed as solutions in a group decision-making problem; ProCLAIM uses Dung's acceptability criteria [Dung, 1995] to determine the winning arguments among the set of all proposed ones; and the AMAL framework follows an informal argumentation theory where cases are used to create the different argument types of the framework and also to evaluate the the preferred ones.

Finally, regarding the main assumptions made by the frameworks, PERSUADER and HERMES mostly rely on the existence of human agents that interact with the system. In the case of PERSUADER, this fact is probably due to the time when the framework was developed, when multi-agent systems were still in their early beginnings. In HERMES' case, the focus of the system on developing a web interface to support group decision-making and not on replacing human judgement motivates this strong assumption on the existence of human users. In addition, most frameworks assume collaborative or cooperative agents and do not tackle with reputation and trust issues. In fact, none of the reviewed frameworks takes into account the possibility to allow malicious and interested agents to come into the system. In this case, the honesty and cooperative predisposition of these agents cannot be assured. The agents of Sycara's PERSUADER system are not cooperative by default, but honesty and rationality in their actions are still assumed. In the ProCLAIM model, the mediator agent can accede

to the argument source manager to evaluate the arguments submitted by an agent in view of its reputation. However, the concept of reputation here seems to be a predefined value for each agent that stands for its expertise in a specific domain. Again, malicious behaviours are not prevented. Therefore, all these frameworks are not intended for operating in open environments with heterogeneous and possibly unreliable agents.

2.6 Open Research Issues

According to the bibliographical analysis performed throughout this chapter, this section introduces open research issues to perform argumentation in agent societies, following a CBR methodology to implement the agents' reasoning capabilities. As pointed out in Chapter 1, in open MAS tracking the arguments that agents put forward in argumentation dialogues and performing a CBR cycle over them could be relatively simple. On the contrary, the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. Thus, a CBR approach to reason about arguments seems to be more suitable than a rule-based approach.

First, the challenge of applying argumentation in agent societies is introduced. This is a new area of research that has to be further investigated. In the previous section, different CBR systems for argumentation have been analysed, trying to identify their important contributions and the research challenges that are dealt with. Then, based on this analysis, we have specified several broad areas that the research community has to further investigate. This section poses open issues in these areas that must be tackled in order to promote the work on argumentation in open MAS with CBR capabilities. Theses open issues offer a wide range of research possibilities to produce interesting advances in CBR, argumentation and open MAS. However, they are not intended to be a comprehensive list of all possible applications of CBR in the wide field of multi-agent argumentation systems, but a set of purposes for promoting new research in the area that inspire the development of this PhD work. Finally, the section provides a conclusion for the state of the art review.

2.6.1 Argumentation in Agent Societies

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

This is the case of [Ferber et al., 2004], which points out some of the drawbacks of classical 'agent centered multi-agent systems'. To resolve these difficulties the authors propose a set of principles and an example methodology to design organization centered multi-agent systems. Also, [Oliva et al., 2008] combines multi-agent argumentation with the agents and artifacts meta-model to design an argumentation component based on Dung's preferred semantics [Dung, 1995]. This component manages arguments and provides a coordination service for argumentation processes in a MAS.

Other works are focused on the study of argumentation in social networks, with a focus on the network structure (or the structure of the group) rather than in the actual social dependencies between software agents or human users. An example of this type is the work presented in [Ontañón and Plaza, 2009], which investigates how argumentation processes among a group of agents may affect the outcome of group judgments in prediction markets. Also, a report on how argumentation can enhance dialogues in social networks can be found in [Heras et al., 2009a].

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

2.6.2 Case-based Argument Representation

A fundamental decision to make in the design of a CBR system that will determine its final operation is the case structure and contents. Bylander and Chandrasekaran stated the *interaction problem* [Bylander and Chandrasekaran, 1987] by which:

'...representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem...'

In the argumentation domain, how to reason about arguments, how to interpret them and how to represent their relations are key issues. A CBR system which purpose is to perform argumentation tasks in MAS must facilitate the reasoning in this domain. Therefore, the underlaying semantics of argumentation dialogues must be taken into account when deciding the structure and the representation language of the cases.

If the arguments that have been submitted in a dialogue are only stored as data in cases of the type *attribute-value*, or similarly, if the cases are simple structures that store information that will be used later to generate arguments, the semantic knowledge acquired during the argumentation is lost. This knowledge could be crucial to develop argumentation strategies able to persuade, for instance, agents with specific profiles or to enhance the interaction protocol (see Section 2.6.3). As pointed out in Section 2.5.6, PERSUADER and Soh's frameworks made the first steps to devise argumentation strategies from the information stored in the cases. In both frameworks, cases store information about the context of previous argumentation processes (e.g. participants' objective, environmental data, negotiation issues, final agreement, etc.). However, the structure and representation language of the cases do not allow to define semantic relations between such concepts.

In addition, if general argumentation knowledge is stored in the cases together with the information about previous argumentation processes (arguments or information pieces to generate them), this knowledge could ease its later interpretation. Our suggestion is to use *Knowledge-Intensive CBR* [Aamodt, 2004], a specific type of CBR methodology that would allow agents to reason with semantic knowledge in addition to the syntactic properties of cases. A possible way to use KI-CBR to argue in open MAS is following an ontological approach to develop systems of this type [Diaz-Agudo and Gonzalez-Calero, 2007]. In this way, ontologies can be used as case representation languages that integrate general terminological knowledge about CBR, argumentation and specific application domains. Further research to specify how to store arguments and dialogues

in the cases of a KI-CBR system and how to reason with them must be carried out. Moreover, an intensive study to determine which type of ontology would be suitable for describing the argumentation concepts in the cases must be performed.

Another challenge that must be dealt with is how to communicate arguments between the agents of a particular MAS or between the agents of different systems. When working with open MAS, where the system dynamicity and the heterogeneity between agents is assumed by default, this functionality is particularly challenging. The research in this area has already been started by the ASPIC community, which is developing its standardisation proposal for an argument interchange format (AIF) [Chesñevar et al., 2006]. The format introduces an abstract formalism for representing concepts about arguments, argument networks, communication and argumentation context in MAS capable of argumentation-based reasoning. Since the AIF is being agreed upon the argumentation and MAS expert research communities, it is likely to be adopted by many researchers as an standard for argument communication. Therefore, considering this proposal when deciding how to represent the syntax and semantics of the cases in a case-based argumentation framework would be advisable for preserving compatibility with other frameworks.

2.6.3 CBR Roles in the Argumentation Process

CBR-based Argument Generation, Selection and Evaluation

Depending on the contents of the case-base, the CBR methodology can play different roles in the argumentation process. The most obvious role (and the one that appears in all frameworks analysed in this research) is to use the case-based reasoning cycle to perform the entire or part of the argument management process (generate, select and evaluate arguments). However, this process has been implemented by using a wide range of techniques without any standardisation in each current approach (see Table 2.2). Furthermore, the selection process in almost all frameworks (except for the AMAL framework) relies on domain-dependent rules and pre-established preference relations. This fact makes increasingly difficult to compare and evaluate the strengths and weaknesses of the CBR-based frameworks for argumentation in MAS.

Over the last years, explanation techniques that make the CBR solving process more understandable have gained an increasing interest [Sørmo et al., 2005]. If case-based arguments are conceived as explanations that justify the position of an agent in a

discussion, the explanation techniques developed in CBR systems could be a good alternative to standardise the generation, selection and evaluation of arguments in MAS. A preliminary work in this direction has been developed in the AMAL framework with the use of LID to generate justified predictions about the class of new problems, but it only applies to classification domains. Also, in the HERMES framework the case-base of the system is used to generate warrants that support the arguments asserted by the users. However, these cases only provide additional information that helps the user to select the best argument to instantiate and do not provide a formal explanation for the generation of specific arguments. Note that anyway, the user is who finally decides which argument poses at any step of the argumentation dialogue, no matter which is the recommendation provided by HERMES. In the rest of frameworks, the only explanation (or justification) for generating particular arguments among the set of all possibilities is the similarity between the current and the previous argumentation dialogues.

Other interesting application of explanations could be that the recipient agent uses them to check the correctness of the argument generation process when arguments are generated from the contents of the sender agent's case-base. In that way, these explanations could act as warrants for their associated arguments and ease the argument evaluation process for the recipient agent.

Besides that, if a knowledge intensive approach is used to store arguments (or pieces of information to generate arguments) in the agent's case-base, the implicit semantic knowledge of the relations between arguments could be also considered in the argument management process. From our point of view, there is a need for an intensive research that specifies how to apply explanation techniques for argument generation, selection and evaluation in different types of argumentation dialogues.

CBR-based Argument Strategies

As stated in [Rahwan et al., 2003], bibliography about dialogue strategies in argumentation is hardly found. The few theoretic research performed to date in the area of argumentation in MAS was reviewed in Section 2.4 and follows two differentiated approaches, the heuristic and the game-theoretic. Other interesting role that the CBR methodology can play in argumentation processes in MAS is to generate heuristic argumentation strategies based on previously acquired experience. Note that one of the main advantages of using CBR to manage argumentation in MAS is that it allows

agents to learn from the process. In the applied research performed in each framework studied in this chapter, the underlying information about the current argumentation dialogue is partially stored in the form of cases when the process finishes. In addition, the agents of the AMAL framework can also learn during the argumentation dialogue by storing in their case-bases the cases that they receive from other agents. However, they do not learn how to predict the behaviour of particular agents or the expected development of the dialogue, but only increase their own knowledge with the knowledge that other agents share with them.

CBR could also be used to learn agents' profiles and generate arguments to perform a strategic argumentation that would easily persuade specific types of agents. Some preliminary steps in this way have already been taken. The fist attempt to use CBR to provide information for building agents' profiles was performed in PERSUADER. In this framework the mediator agent uses the information about previous negotiation processes stored in the case-base to develop the behavioural model of an unknown agent and devise the best way to persuade it. Similarly, in Soh's framework the information of the cases is used to decide which type of arguments are best suited to convince an agent with a specific profile and to infer other parameters that influence the negotiation process (e.g. time constraints, resources usage, etc.). Nevertheless, in both cases the argumentation strategy is highly domain dependent and completely relies on previous knowledge. Although in PERSUADER the agents' models can be dynamically updated, the preference order that determines which argument must be ultimately posed depends on a pre-established hierarchy.

In a more dynamic and online way, the case-base could be used to store information about the agents' profile that could be gathered either by observing the current agents' behaviour, by learning information that the agents send during the dialogue or as a result of inquiry and information seeking processes. Therefore, this information could be used in the current argumentation process to generate and select better arguments to put forward and to evaluate the incoming ones. Case-based argumentation strategies have to be further investigated and there is still much work to do in this area.

CBR as Guide of the Interaction Protocol

The interaction protocol between the agents of a MAS needs rules to govern the dialogue (e.g. the agents' entries and withdraws from the dialogue, the validity of proposals, the fulfilment of commitments and the termination of the dialogue). Our view is that

CBR can play an important role as a useful tool to define such rules. For instance, the acceptable time to terminate an argumentation dialogue could be inferred from the information stored in the case-base about past similar arguments that ended in disagreement due to their excessive duration. On the other way round, the time to reach an agreement could be inferred from the time that took to reach similar agreements in the past.

When defining the current negotiation strategy, the Soh's framework already considers information about the time, number of steps and resources usage in previous negotiation processes. However, as pointed out before, the cases are completely situated and to be applicable in current negotiations they need to be adapted by using domain dependent adaption rules. In addition, only one case (the one that is potentially most similar to the current situation) is used to define the negotiation strategy. Our purpose is to develop algorithms that take into account the argumentation parameters of not only one specific case, but of a set of previous similar cases.

In addition, if CBR is used to generate and evaluate arguments, infinite dialogues can be avoided by introducing certain rules in the reasoning cycle. The AMAL framework, for example, does not allow an agent to generate the same case-based argument twice during a dialogue. This is a basic strategy, but still very effective. More elaborated rules could avoid circular arguments and prevent the introduction of fallacies that could deviate the argumentation dialogue from its main objective of reaching an agreement.

Finally, CBR could warrant the argumentation success by stopping the process or modifying certain parameters when the agents notice that the current dialogue is developing in a similar way to other precedent that found insuperable obstacles to reach a profitable agreement.

Note that the case-based guide of the interaction protocol could also be considered as another type of argumentation strategy and then, it should be included in the previous section. For clarity purposes, since this strategy is focused on the interaction process itself and not on the argumentation management, we have decided to study it in this separate section.

2.6.4 Case-base Consistency Matters

An important open issue that we have identified and that has received little attention in the literature is how to update arguments that were generated from past experiences

to fit current argumentation dialogues. The case-base consistency is a critical issue to ensure the proper operation of a CBR system along the time. In the dynamic context of an open dialogue, where the agents can enter as new participants or well finish the interaction with the other agents, how a change of this type (environmental, in the agents' points of view, etc.) can affect to the validity of the case-base information must be considered. Otherwise, the arguments inferred from the case-base could become obsolete.

To ensure consistency, powerful algorithms to adapt and maintain the case-base must be implemented. Such algorithms must be able to adapt situated cases and make them applicable to the current problem or otherwise, eliminate them. Soh's model deals with this functionality to a certain extent, by using domain-specific adaption rules. However, due to the dynamism of the argumentation domain, such rules can quickly become obsolete. Therefore, the adaption methods must be as domain-independent as possible.

2.6.5 Trust and Reputation

None of the models that have been studied in this paper have taken reputation and trust issues into account as tools for preventing the system from unintentionally wrong or malicious behaviours. If the case-based argumentation framework is conceived to operate in open MAS these functionalities are important. Before storing any information in the case-base during an argumentation dialogue between several agents, an agent must check the trustworthiness of such information. The opposite process is also necessary. If the profile that an agent has about other agent changes, resulting in a decrease of the the other's reputation, the information that comes from the interactions with this untrustworthy agent stored in the former's case base must be revised. Other important question that needs from further research is whether an agent must trust the information that it generates from its case-base by default. Trust and reputation issues must not be underestimated in argumentation frameworks in open MAS environments.

2.7 Conclusions

This chapter has introduced several concepts of argumentation theory that have been applied to model the reasoning and behaviour of agents in MAS. Among them, we

use dialogue games, commitment stores and argumentation schemes as elements of the framework proposed in this PhD thesis. Argumentation in AI has historically followed two different approaches based on rule-based or case-based systems. The dynamics of open MAS makes difficult to specify in advance the set of rules that govern the behaviour of agents. Thus, a case-based approach to track arguments exchanged among agents and learn from the experience is a most suitable option to perform argumentation in agent societies. Therefore, we have focused the state of the art analysis on the few research approaches on case-based argumentation in MAS proposed to date.

The literature review shows that a formal and context-independent framework that defines an argumentation theory for agents with learning capabilities that are situated in a society and have values to promote/demote does not already exist. This motivates the development of the framework proposed in the next chapter. The examples about systems that successfully apply CBR to manage argumentation in MAS demonstrate the suitability of this reasoning methodology to provide agents with the ability to argue. However, the current approaches are domain dependent and some assume the interaction of humans with the system. Therefore, they are not suitable to be applied in a general domain where agents argue in the context of a society in a MAS. Also, none of the systems reviewed has the ability of managing agents' values and dependency relations in the argumentation process. This ability is crucial in agent societies, where these are important concepts of the social context of agents.

As will be shown in Chapters 3 and 4, the case-based argumentation framework proposed in this thesis makes an innovative contribution on some of the open issues identified in this chapter. Concretely, our framework is addressed to agent societies and the representation, generation, selection and evaluation of arguments is completely case-based. Also, by taking into account previous argumentation knowledge stored in the form of cases, agents can follow different argumentation strategies.

Part III

Proposal

Chapter

3

Proposed Framework

3.1 Introduction

In this chapter we propose a computational framework for design and implementation of MAS in which the participating software agents are able to manage and exchange arguments between themselves taking into account the agents' social context. First, we propose a formal definition for an agent society and analyse the necessary requirements for an argumentation framework for agent societies. After that, we introduce the knowledge resources that agents can use to manage their positions and arguments. The knowledge resources of the framework and other argumentation related concepts are defined by using an OWL-DL ontology called ArgCBROnto. Also, we follow a case-based approach to represent arguments. Thus, our knowledge resources are designed as cases with the common structure of cases in CBR systems.

Then, we provide an abstract argumentation framework for agent societies by extending abstract value-based argumentation frameworks to work with agent societies. After that, the case-based argumentation framework proposed is instantiated by defining the structure and relations of its elements. Following, the reasoning process by which agents can automatically generate, select and evaluate arguments is presented. This process allows agents to learn from argumentation experiences and makes easier to develop dialogue strategies.

3.2 Requirements for an Argumentation Framework for Agent Societies

In this section, we introduce the formal definition of the concepts that define our approach for agent societies. Then, we analyse the issues that have been considered to choose a suitable argumentation framework for agent societies.

3.2.1 Society Model

In this thesis, we follow the approach of [Dignum, 2003] and [Artikis et al., 2009], who define an *agent society* in terms of a set of *agents* that play a set of *roles*, observe a set of *norms* and a set of *dependency relations* between roles and use a *communication language* to collaborate and reach the global objectives of the *group*. This definition is generic enough to fit most types of agent societies, such as social networks of agents or open agent organisations. Broadly speaking, it can be adapted to any open MAS where there are norms that regulate the behaviour of agents, roles that agents play, a common language that allow agents to interact defining a set of permitted locutions and a formal semantics for each of these elements. Here, we differentiate the concept of agent society from the concept of group of agents, in the sense that we consider that the society is the entity that establishes the dependency relations between the agents, which can dynamically group together to pursue common objectives in a specific situation. Moreover, the set of norms in open MAS define a *normative context* (covering both the set of norms defined by the system itself as well as the norms derived from agents' interactions)[Criado et al., 2009].

However, we consider that the values that individual agents or groups want to promote or demote and preference orders over them have also a crucial importance in the def-

inition of an argumentation framework for agent societies. These values could explain the reasons that an agent has to give preference to certain beliefs, objectives, actions, etc. Also, dependency relations between roles could imply that an agent must change or violate its value preference order. For instance, an agent of higher hierarchy could impose its values to a subordinate or an agent could have to adopt a certain preference order over values to be accepted in a group. Therefore, we endorse the view of [Perelman and Olbrechts-Tyteca, 1969], [Searle, 2001] and [Bench-Capon and Atkinson, 2009], who stress the importance of the audience in determining whether an argument (e.g. for accepting or rejecting someone else's beliefs, objectives or action proposals) is persuasive or not. Thus, we have included in the above definition of agent society the notion of values and preference orders among them. Next, we provide a formal definition for the model of society that we have devised:

Definition 3.2.1 (Agent Society) *An Agent society in a certain time t is defined as a tuple* $S_t = < Ag, Rl, D, G, N, V, Roles, Dependency, Group, val, Valpref_Q >$ *where:*

- $Ag = \{ag_1, ag_2, ..., ag_I\}$ *is a finite set of* I *agents members of* S_t *in a certain time* t.
- $Rl = \{rl_1, rl_2, ..., rl_J\}$ *is a finite set of* J *roles that have been defined in* S_t.
- $D = \{d_1, d_2, ..., d_K\}$ *is a finite set of* K *possible dependency relations among roles defined in* S_t.
- $G = \{g_1, g_2, ..., g_L\}$ *is a finite set of groups that the agents of* S_t *form, where each* $g_i, 1 \leq i \leq L, g_i \in Ag$ *consist of a set of agents* $a_i \in A$ *of* S_t.
- N *is a finite set of norms that affect the roles that the agents play in* S_t.
- $V = \{v_1, v_2, ..., v_P\}$ *is a set of* P *values predefined in* S_t.
- $Roles : Ag \rightarrow 2^{Rl}$ *is a function that assigns an agent its roles in* S_t.
- $Dependency_{S_t} : \forall d \in D, <_d^{S_t} \subseteq Rl \times Rl$ *defines a reflexive, symmetric and transitive partial order relation over roles.*
- $Group : Ag \rightarrow 2^G$ *is a function that assigns an agent its groups in* S_t.
- $val : Ag \rightarrow V$ *is a function that assigns an agent the set of values that it has.*

- $Valpref_Q \subseteq V \times V$, *where* $Q = Ag \vee Q = G$, *defines a reflexive, symmetric and transitive partial order relation* $<_Q^{S_t}$ *over the values of an agent of a group.*

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

3.2.2 Computational Requirements for Arguments in Agent Societies

An argumentation process is conceived as a reasoning model with several steps:

1. Building arguments (supporting or attacking conclusions) from knowledge bases.
2. Defining the strengths of those arguments by comparing them in conflict situations.
3. Evaluating the acceptability of arguments in view of the other arguments that are posed in the dialogue.
4. Defining the justified conclusions of the argumentation process.

The first step to design MAS whose agents are able to perform argumentation processes is to decide how agents represent arguments. According to the *interaction problem* defined in [Bylander and Chandrasekaran, 1987], *"...representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem..."*. Therefore the way in which agents computationally represent arguments should ease the automatic performance of argumentation processes.

Most research effort on the computational representation of arguments is performed in the area of developing models for argument authoring and diagramming [Rahwan et al., 2007b][Rowe and Reed, 2008](OVA[1]). However, these systems assume human users interacting with the software tool and are not conceived for performing agents' automatic reasoning processes. Other research works where the computational modelling of arguments has been studied are those on case-based argumentation. From

the first uses of argumentation in AI, arguments and cases are intertwined [Skalak and Rissland, 1992]. As pointed out in Chapter 2, case-based argumentation particularly reported successful applications in American common law [Bench-Capon and Dunne, 2007], whose judicial standard orders that similar cases must be resolved with similar verdicts. In [Bench-Capon and Sartor, 2003] a model of legal reasoning with cases is proposed. But, again, this model assumed human-computer interaction and cases were not thought to be only acceded by software agents. Case-Based Reasoning (CBR) systems [Aamodt and Plaza, 1994] allow agents to learn from their experiences. In MAS, the research in case-based argumentation is quite recent with just a few proposals to date. These proposals are highly domain-specific or centralise the argumentation functionality in a *mediator* agent that manages the dialogue between the agents of the system [Heras et al., 2009b].

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

Taking into account the above issues, there are a set of requirements that a suitable framework to represent arguments in agent societies should meet:

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

These requirements suggest that an argumentation framework for agent societies should be easily interpreted by machines and have highly expressive formal semantics to define complex concepts and relations over them. Thus, we propose a Knowledge-Intensive (KI) case-based argumentation framework [Diaz-Agudo and Gonzalez-Calero, 2007], which allows automatic reasoning with semantic knowledge in addition to the syntactic properties of cases. Reasoning with cases is specially suitable when there is a weak (or even unknown) domain theory, but acquiring examples encountered in practice is easy. Most argumentation systems produce arguments by applying a set of inference rules. In open MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. However, tracking the arguments that agents put forward in argumentation processes could be relatively simple. Other important problem with rule-based systems arises when the knowledge-base must be updated (e.g. adding new knowledge that can invalidate the validity of a rule). Updates imply to check the knowledge-base for conflicting or redundant rules. Case-based systems are in most cases easier to maintain than rule-based systems and hence, more suitable for being applied in dynamic domains.

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

3.3 Knowledge Resources

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

- Domain-cases database, with domain-cases that represent previous problems and their solutions. The structure of these cases is domain-dependent and thus is not detailed in this chapter.
- Argument-cases database, with argument-cases that represent previous argumentation experiences and their final outcome.
- Argumentation schemes [Walton et al., 2008], with a set of argumentation schemes,

which represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. Also, each argumentation scheme has associated a set of *critical questions* that represent potential attacks to the conclusion supported by the scheme. The concrete argumentation schemes to be used depend on the application domain.

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

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

In this section we describe the above knowledge resources by using an ontological approach that observes the requirements put forward in the last section. Thus, this ontology uses Description Logics (DLs) to provide a common language to represent the resources that is computationally tractable, rich enough to represent different types of domain-specific and general knowledge, generic enough to represent different types of arguments and compliant with the technological standards of data and argument interchange in the Web.

Description Logics (DLs) are a family of formal knowledge representation languages that are used in AI for formal reasoning on the concepts of an application domain (terminological knowledge). In DL the knowledge base consist of a set of terminological axioms (or $TBox$) that contains sentences describing relations between concepts and a set of assertional axioms (or $ABox$) that describes the relations between individuals and concepts (where in the hierarchy of concepts the individuals belong). Thus, DL distinguishes between concepts, roles, which are properties of these concepts and individuals, which are instances of the concepts. Table 3.1 shows the syntax and interpretation of the DL definitions provided in this paper. In the table, C and D are concepts, R is a role, a and b are individuals and n is a positive integer.

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

Table 3.1: DL Notation

3.3.1 ArgCBROnto Ontology: General Concepts

We have designed an ontology called *ArgCBROnto* to define the representation language of the above knowledge resources. Thus, the vocabulary of domain-cases, argument-cases and argumentation schemes is defined by using this ontology, which follows the approach of the case-based KI approach proposed in [Diaz-Agudo and Gonzalez-Calero, 2007] and the Argument Interchange Format (AIF) ontology [Willmott et al., 2006]. KI-CBR enables automatic reasoning with semantic knowledge in addition to the syntactic properties of cases. This allows making semantic inferences with the elements of cases and using more complex measures to compute the similarity between them. In addition, the AIF ontology provides a common vocabulary to represent argumentation concepts between different argumentation frameworks. In this sense, AIF provides an abstract representation framework for argumentation systems while our ArgCBROnto is designed to facilitate case-based reasoning in an argumentation framework designed to be used by agents societies. However, both formalisms are compatible, as will be discussed in Section 3.7. Next, we provide a general view of the ArgCBROnto ontology for the argumentation framework proposed in this chapter, with focus on the concepts that define the knowledge resources presented in this section.

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

$Case \sqsubseteq Thing \quad Case \sqsubseteq \neg CaseComponent$

$CaseComponent \sqsubseteq Thing \quad CaseComponent \sqsubseteq \neg ArgumentationScheme$

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

As pointed out before, there are two disjoint types of cases, *domain-cases* and *argument-cases* (see Figure 3.1).

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

$ArgumentCase \sqsubseteq \neg DomainCase$

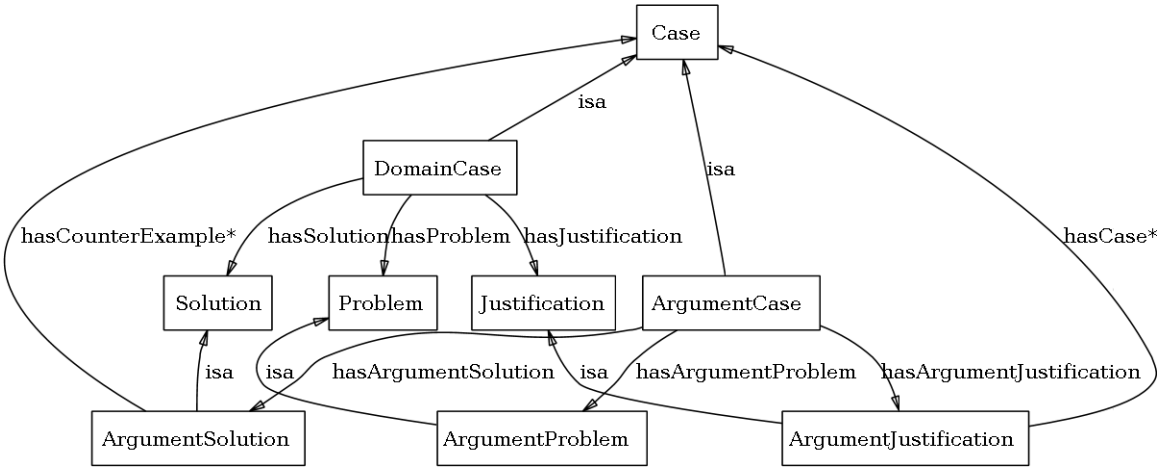

Figure 3.1: ArgCBROnto Case

Cases have the three possible types of components that usual cases of CBR systems have (see Figure 3.2): the description of the state of the world when the case was stored (*Problem*); the solution of the case (*Conclusion*); and the explanation of the process that gave rise to this conclusion (*Justification*). These concepts are disjoint.

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

$Justification \sqsubseteq CaseComponent$

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

$Problem \sqsubseteq \neg Justification$

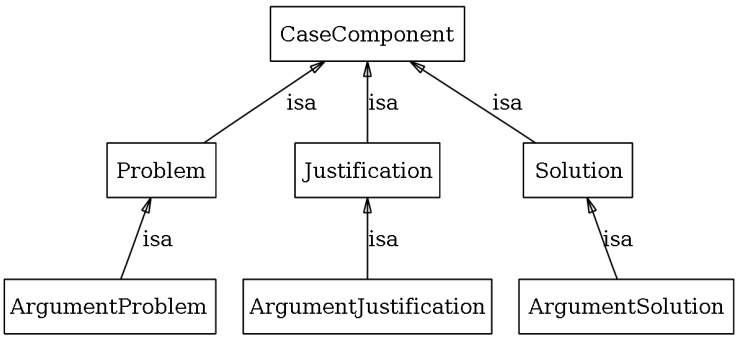

Figure 3.2: ArgCBROnto CaseComponent

Domain-cases have the usual problem, conclusion and justification parts, as shown in Figure 3.1.

$DomainCase \sqsubseteq \forall hasProblem.Problem$

$DomainCase \sqsubseteq \exists hasProblem.Problem$

$DomainCase \sqsubseteq = 1 hasProblem$

$DomainCase \sqsubseteq \forall hasSolution.Solution$

$DomainCase \sqsubseteq \exists hasSolution.Solution$

$DomainCase \sqsubseteq \geq 1 hasSolution$

$DomainCase \sqsubseteq \forall hasJustification.Justification$

$DomainCase \sqsubseteq \leq 1 hasJustification$

However, argument-cases have a more specialised description for these components (*ArgumentProblem*, *ArgumentSolution* and *ArgumentJustification*), which includes an extended set of properties (see Figure 3.1).

$ArgumentProblem \sqsubseteq Problem \; ArgumentSolution \sqsubseteq Solution$

$ArgumentJustification \sqsubseteq Justification$

$ArgumentCase \sqsubseteq \forall hasArgumentProblem.ArgumentProblem$

$ArgumentCase \sqsubseteq \exists hasArgumentProblem.ArgumentProblem$

$ArgumentCase \sqsubseteq= 1 hasArgumentProblem$

$ArgumentCase \sqsubseteq \forall hasArgumentSolution.ArgumentSolution$

$ArgumentCase \sqsubseteq \exists hasArgumentSolution.ArgumentSolution$

$ArgumentCase \sqsubseteq= 1 hasArgumentSolution$

$ArgumentCase \sqsubseteq \forall hasArgumentJustification.ArgumentJustification$

$ArgumentCase \sqsubseteq \exists hasArgumentJustification.ArgumentJustification$

$ArgumentCase \sqsubseteq= 1 hasArgumentJustification$

Also, cases have as properties a unique identifier *ID* and a *creation date*, with its corresponding range and domain. Note that these properties have as domain several concepts of the ArgCBROnto ontology which will be introduced later.

$\top \sqsubseteq \forall hasID.Integer$

$\top \sqsubseteq \forall\ hasID^{-}$.(Case $\sqcup$ SocialEntity $\sqcup$ Value $\sqcup$ Norm $\sqcup$ Argument $\sqcup$ Argumentation-Scheme $\sqcup$ Premise)

$\top \sqsubseteq \forall$ hasCreationDate.Date $\top \sqsubseteq \forall hasCreationDate^{-}$.(Case $\sqcup$ Argumentation-Scheme)

As pointed out before section, argumentation schemes represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. Each argumentation scheme consists of a set of *premises*, a *conclusion* drawn from these premises and a set of *critical questions* that represent potential attacks to the conclusion supported by the scheme. These critical questions can be classified as *presumptions* that the proponent of the argumentation scheme has made or *exceptions* to the general inference rule that the scheme represents [Prakken et al., 2005]. In the former case, the proponent has the burden of proof if the critical question is asked, whereas in the later the burden of proof falls on the opponent that has questioned the conclusion of the scheme. Figure 3.3 shows the representation of an argumentation scheme in the ArgCBROnto ontology.

$ArgumentationScheme \sqsubseteq Thing$

$ArgumentationScheme \sqsubseteq \forall hasPremise.Premise$

$ArgumentationScheme \sqsubseteq \exists hasPremise.Premise$

$ArgumentationScheme \sqsubseteq \geq 1 hasPremise$

$ArgumentationScheme \sqsubseteq \forall hasConclusion.Conclusion$

$ArgumentationScheme \sqsubseteq \exists hasConclusion.Conclusion$

$ArgumentationScheme \sqsubseteq = 1 hasConclusion$

$ArgumentationScheme \sqsubseteq \forall hasPresumption.Premise$

$ArgumentationScheme \sqsubseteq \forall hasException.Premise$

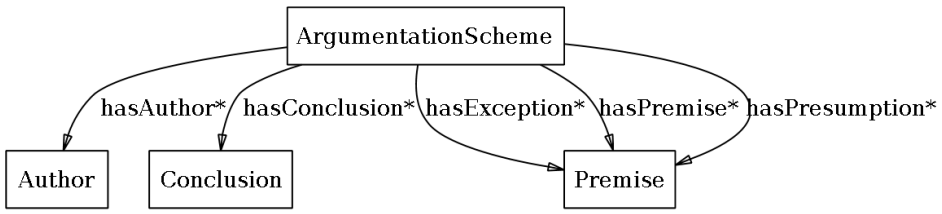

Figure 3.3: ArgCBROnto ArgumentationScheme

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

$\top \sqsubseteq \forall argTitle.String$

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

$\top \sqsubseteq \forall creationDate.Date$

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

$ArgumentationScheme \sqsubseteq \forall hasAuthor.Author$

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

3.3.2 Argument-case Description

Argument-cases have two main objectives:

1. They can be used by agents as knowledge resources to generate new arguments and to select the best position to put forward in view of past argumentation experiences.

2. They can be used to store new argumentation knowledge that agents gain in each dialogue, improving the agents' argumentation skills.

Table 3.2 shows the structure of a generic argument-case. As pointed out before, the argument-cases have three main parts: the description of the *problem* that the case represents, the *solution* applied to this problem and the *justification* why this particular solution was applied. An argument-case stores the information about a previous argument that an agent posed in certain step of a dialogue with other agents.

Problem:

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

$Context \sqsubseteq Thing$

$DomainContext \sqsubseteq Context$

$Problem \sqsubseteq \forall hasDomainContext.DomainContext$

$Problem \sqsubseteq \exists hasDomainContext.DomainContext$

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

Table 3.2: Structure of an Argument-Case.

$Problem \sqsubseteq = 1 hasDomainContext$

$Premise \sqsubseteq Thing$

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

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

In addition, if we want to store an argument and use it to generate a persuasive argument in the future, the features that characterise the audience of the previous argument (the *social context*) must also be kept. Thus, we have two disjoint types of contexts in our ontology, the usual domain context and the social context (shown in Figure 3.4).

$SocialContext \sqsubseteq Context$

$DomainContext \sqsubseteq \neg SocialContext$

$ArgumentProblem \sqsubseteq \forall hasSocialContext.SocialContext$

$ArgumentProblem \sqsubseteq \exists hasSocialContext.SocialContext$

$ArgumentProblem \sqsubseteq= 1 hasSocialContext$

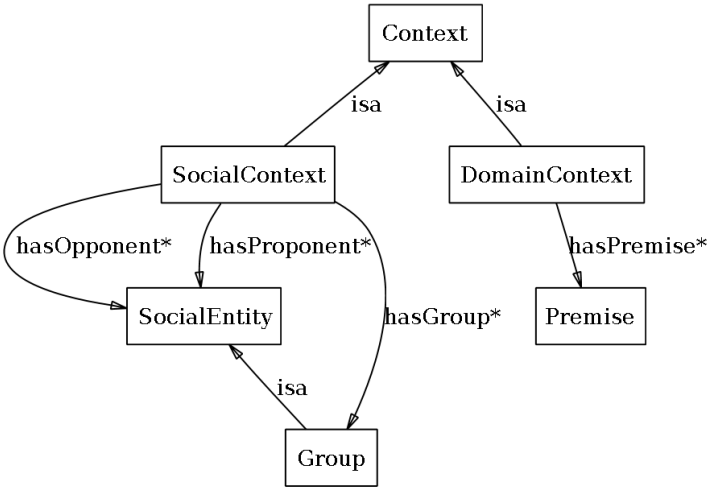

Figure 3.4: ArgCBROnto Context

For the definition of the social context of arguments, we follow our model of society presented in Section 3.2.1. Therefore, we store in the argument-case the social information about each *social entity* related with the argument. This social entity can be an *agent* (the proponent of the argument and the opponent to which the argument is addressed) or else the *group* to which both agents belong. Figure 3.5 shows this part of the ArgCBROnto ontology.

$SocialEntity \sqsubseteq Thing$

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

$Agent \sqsubseteq \neg Group$

For the sake of simplicity, in this chapter we assume that in each step of the dialogue, one proponent agent generates an argument and sends it to one opponent agent that belongs to its same group. However, either the proponent or the opponent's features could represent information about agents that act as representatives of a group and

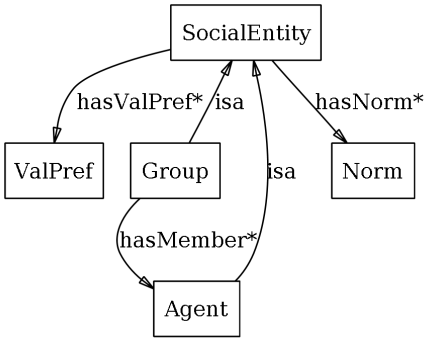

Figure 3.5: ArgCBROnto SocialEntity

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

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

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

$$SocialContext \sqsubseteq \forall hasGroup.Group$$

$$Group \sqsubseteq \forall hasMember.Agent$$

$$Group \sqsubseteq \exists hasMember.Agent$$

$$Group \sqsubseteq \geq 2 hasMember$$

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

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

In addition, for each social entity a reference to the set of *norms* that governed the behaviour of the agents at this step of the dialogue is also stored, since the normative

context of agents could force or forbid them to accept certain facts and the arguments that support them (e.g. a norm could invalidate a dependency relation or a value preference order). Also, each norm has a unique *ID* that identifies it (the ontological definition was provided before) and a *description* with the semantics of the norm.

$Norm \sqsubseteq Thing$

$SocialEntity \sqsubseteq \forall hasNorm.Norm$

$\top \sqsubseteq \forall hasDescription.String$

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

Moreover, if known, we also store the preferences of each agent or group over the predefined set of general *values* in the system (e.g. security, solidarity, economy, etc.). As pointed out before, these preferences (*ValPref*) affect the persuasive power of the proponent's argument over the opponent's behaviour. In the case of the group, we use this feature to store its *social values*[1].

$Value \sqsubseteq Thing$

$ValPref \sqsubseteq Thing$

$ValueNode \sqsubseteq Thing$

$ValueNode \sqsubseteq \forall hasValue.Value$

$ValueNode \sqsubseteq \exists hasValue.Value$

$ValueNode \sqsubseteq = 1 hasValue$

$ValueNode \sqsubseteq \forall hasPrevious.ValueNode$

$ValueNode \sqsubseteq \forall hasNext.ValueNode$

$ValPref \sqsubseteq \forall hasPreferred.Value$

$ValPref \sqsubseteq \exists hasPreferred.Value$

$ValPref \sqsubseteq = 1 hasPreferred$

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

$ValPref \sqsubseteq \forall hasValueNode.ValueNode$

$SocialEntity \sqsubseteq \forall hasValPref.ValPref$

Finally, the *dependency relation* between the proponent's and the opponent's roles is also stored in the social context of the argument-cases. To date, we define the possible dependency relations between roles as in [Dignum and Weigand, 1995]:

- *Power*: when an agent has to accept a request from other agent because of some pre-defined domination relationship between them (e.g. in a society S_t that manages the water of a river basin, $Farmer <^{S_t}_{Power} BasinAdministrator$, since farmers must comply with the laws announced by the basin administrator [2].
- *Authorisation*: when an agent has committed itself to other agent for a certain service and a request from the latter leads to an obligation when the conditions are met (e.g. in the society S_t, $Farmer_i <^{S_t}_{Authorisation} Farmer_j$, if $Farmer_j$ has contracted a service that offers $Farmer_i$).
- *Charity*: when an agent is willing to answer a request from other agent without being obliged to do so (e.g. in the society S_t, by default $Farmer_i <^{S_t}_{Charity} Farmer_j$).

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

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

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

Solution:

In the solution part, the *conclusion* of the case (for both domain-cases and argument-cases) and the *value* promoted in this specific situation are stored (see Figure 3.6).

$Conclusion \sqsubseteq Thing$

$Solution \sqsubseteq \forall hasConclusion.Conclusion$

[2] This example will be explained in detail in Chapter 5.

$Solution \sqsubseteq \exists hasConclusion.Conclusion$

$Solution \sqsubseteq= 1hasConclusion$

$Solution \sqsubseteq \forall promotesValue.Value$

$Solution \sqsubseteq \exists promotesValue.Value$

$Solution \sqsubseteq= 1promotesValue$

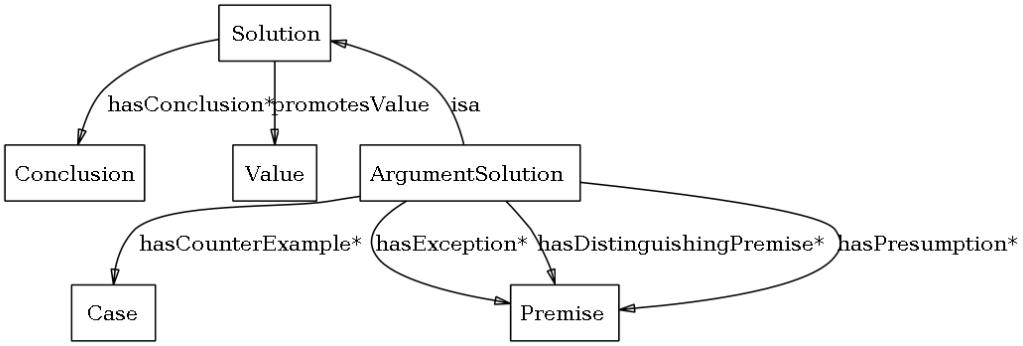

Figure 3.6: ArgCBROnto Solution

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

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

$ArgumentSolution \sqsubseteq Solution$

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

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

Moreover, the solution part of the argument-cases stores the information about the *acceptability status* of the argument at the end of the dialogue. This feature shows if the argument was deemed *acceptable*, *unacceptable* or *undecided* in view of the other arguments that were put forward during the dialogue (see Section 3.5 for details).

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

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

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

- For presumptive arguments: critical questions (*presumptions* or *exceptions*) associated with the scheme [Walton et al., 2008].
- For inductive arguments, as proposed in [Bench-Capon and Sartor, 2003], either:
  - Premises which value in the context where the argument was posed was different (or non-existent) than the value that it took in the cases used to generate the argument (*distinguishing premises*) or
  - Cases which premises also match the premises of the context where the argument was posed, but which conclusion is different than the conclusion of the case(s) used to generate the argument (*counter-examples*).
- For mixed arguments: any of the above attacks.

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

$ArgumentSolution \sqsubseteq \forall hasPresumption.Premise$

$ArgumentSolution \sqsubseteq \forall hasException.Premise$

$ArgumentSolution \sqsubseteq \forall hasDistinguishingPremise.Premise$

$ArgumentSolution \sqsubseteq \forall hasCounterExample.Case$

Justification:

In the ArgCBROnto ontology, the justification part of a case stores a *description* that can explain why this particular solution was applied to solve the case and what was the final results achieved. Figure 3.7 shows the concepts of the ArgCBROnto ontology that are related with this part of argument-cases.

$\top \sqsubseteq \forall hasDescription.String$

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

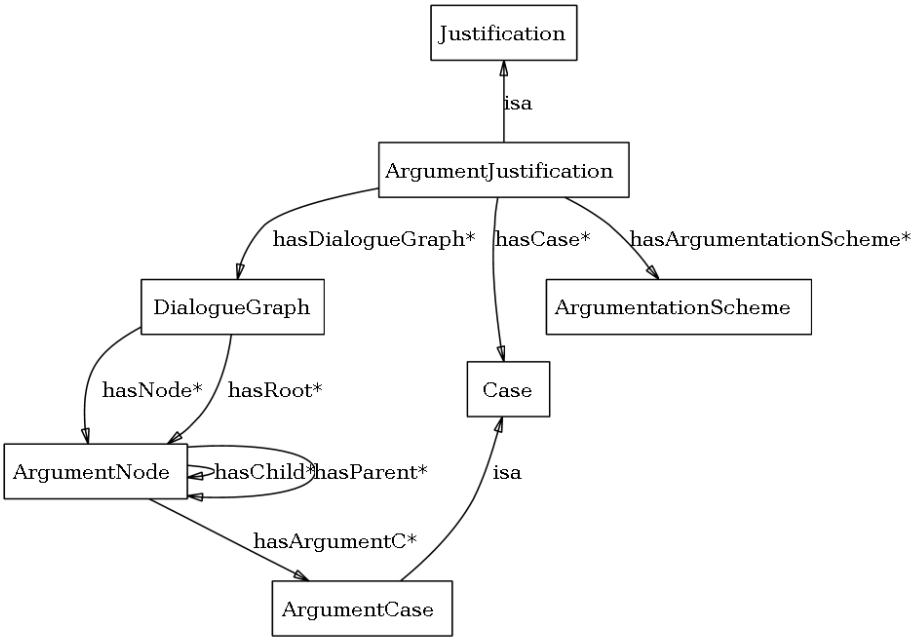

Figure 3.7: ArgCBROnto Justification

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

schemes in presumptive arguments, the set of *cases* in inductive arguments and both in mixed arguments).

$ArgumentJustification \sqsubseteq Justification$

$ArgumentJustification \sqsubseteq \forall\ hasArgumentationScheme.ArgumentationScheme$

$ArgumentJustification \sqsubseteq \forall hasCase.Case$

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

$DialogueGraph \sqsubseteq Thing$

$ArgumentNode \sqsubseteq Thing$

$ArgumentNode \sqsubseteq \forall hasArgumentC.ArgumentCase$

$ArgumentNode \sqsubseteq \exists hasArgumentC.ArgumentCase$

$ArgumentNode \sqsubseteq= 1hasArgumentC$

$ArgumentNode \sqsubseteq \forall hasParent.ArgumentNode$

$ArgumentNode \sqsubseteq \forall hasChild.ArgumentNode$

$DialogueGraph \sqsubseteq \forall hasRoot.ArgumentNode$

$DialogueGraph \sqsubseteq \exists hasRoot.ArgumentNode$

$DialogueGraph \sqsubseteq = 1 hasRoot$

$DialogueGraph \sqsubseteq \forall hasNode.ArgumentNode$

$ArgumentJustification \sqsubseteq \forall hasDialogueGraph.DialogueGraph$

$ArgumentJustification \sqsubseteq \exists hasDialogueGraph.DialogueGraph$

$ArgumentJustification \sqsubseteq \geq hasDialogueGraph$

Following a CBR methodology, the proposed knowledge resources allow agents to automatically generate, select and evaluate arguments. However, the specification of this case-based reasoning process is out of the scope of this chapter. Here we have focused on defining how agents can represent arguments and argumentation related information to be able to perform an efficient and automatic management of this information. The argument-case structure presented is flexible enough to represent different types of arguments and their associated information. Also, the KI approach followed allows a semantic reasoning with the concepts that represent the cases. However, the value of some features on argument-cases and domain-cases could remain unspecified in some domains. For instance, in some open MAS, the preferences over values of other agents could not be previously known, although agents could try to infer the unknown features by using CBR adaptation techniques [López de Mántaras et al., 2006]. This and other open questions will be deal with again in Section 3.7. Once the knowledge resources of our framework and the ArgCBROnto ontology that describes them have been presented in this section, next section provides an abstract definition for the framework.

3.4 Abstract Argumentation Framework for Agent Societies

Most abstract argumentation frameworks (AFs) are based on Dung's framework [Dung, 1995], which is defined as a pair $< A, R >$ where A is a set of arguments and $R \subseteq A \times A$ is a binary *attack* relation on A. For two arguments A and B, $R(A, B)$ means that the argument A attacks the argument B. AF abstract the structure and meaning of arguments and attacks between them and focus their research efforts on analysing generic properties and *argumentation semantics*. This semantics is the formal definition of the method by which arguments are evaluated in view of other arguments [Baroni and Giacomin, 2009]. Semantics can be *extension-based*, which determine the *extensions* or sets of arguments that can be collectively acceptable or *labelling-based*, which label

each argument of A with a specific state in a predetermined set of possible states of an argument.

Based on Dung's AF, we define an Argumentation Framework for an Agent Society ($AFAS$) as:

Definition 3.4.1 (Argumentation Framework for an Agent Society) *An argumentation framework for an agent society is a tuple $AFAS = <A, R, S_t>$ where:*

- *A is a set of arguments.*
- *R is an irreflexive binary attack relation on A.*
- *S_t is a society of agents as defined in Definition 3.2.1.*

Then, we specialise $AFAS$ considering them for an specific agent, since each agent of an open MAS can have a different preference order over values. Thus, an *audience* is defined as a preference order over values. For the definition of our Agent specific Argumentation Framework for Agent Societies we start from the definition of Audience specific Value-based Argumentation Frameworks (AVAF) [Bench-Capon and Atkinson, 2009]. This is also based on Dung's and we will extend and adapt it to take into account the social context of agents.

Definition 3.4.2 (Audience-specific Value-based AF) *An audience-specific value-based argumentation framework is a 5-tuple $AVAF_a = < A, R, V, val, Valpref_a >$ where:*

- *A, R , V and val are as defined for a Value-based Argumentation Framework (VAF) [Bench-Capon and Atkinson, 2009].*
- *$a \in P$ is an audience of the set of audiences P.*
- *$Valpref_a \subseteq V \times V$ is a transitive, irreflexive and asymmetric preference relation that reflects the value preferences of the audience a.*

Then, we extend AVAFs and define our abstract Agent-specific Argumentation Framework in an Agent Society ($AAFAS$) as follows:

Definition 3.4.3 (Agent-specific AF for an Agent Society) *An agent specific argumentation framework for an agent society is a tuple $AAFAS = < Ag, Rl, D, G, N, A, R, V, Role, Dependency_{S_t}, Group, Values, val, Valpref_{ag_i} >$ where:*

- *Ag, Rl, D, G, N, A, R, V, $Dependency_{S_t}$, Group and Values are defined as in Definition 3.2.1.*
- *$Role(ag, a) : Ag \times A \rightarrow Rl$ is a function that assigns an agent the specific role that it plays (from its set of roles) when it has put forward a specific argument.*
- *$val(ag, a) : Ag \times A \rightarrow 2^V$ is a function that assigns an agent's argument the value(s) that it promotes.*
- *$Valpref_{ag_i} \subseteq V \times V$, defines a irreflexive, transitive and asymmetric relation $<^{S_t}_{ag_i}$ over the agent's ag_i values in the society S_t.*

The aim of $AAFAS$ is to determine which agent's argument attacks other agent's argument in an argumentation process performed in a society of agents and in each case, which argument would defeat the other. To do that, we have to consider the values that arguments promote and their preference relation as in AVAFs, but also the dependency relations between agents. These relations could be stronger than value preferences in some cases (depending on the application domain). For the time being, as in [Dignum and Weigand, 1995], we only consider the following dependency relations:

- *Power*: when an agent has to accept a request from other agent because of some pre-defined domination relationship between them. For instance, in a society S_t that manages the water-rights transfer of a river basin, $Farmer <^{S_t}_{Pow} BasinAdministrator$, since farmers must comply with the laws announced by the basin administrator.

- *Authorisation*: when an agent has committed itself to other agent for a certain service and a request from the latter leads to an obligation when the conditions are met. For instance, in S_t, $Farmer_i <^{S_t}_{Auth} Farmer_j$, if $Farmer_j$ has contracted a service that offers $Farmer_i$.

- *Charity*: when an agent is willing to answer a request from other agent without being obliged to do so. For instance, in S_t, by default $Farmer_i <^{S_t}_{Ch} Farmer_j$ and $Farmer_j <^{S_t}_{Ch} Farmer_i$.

Thus, we can now define the agent-specific defeat relation of $AAFAS$ as:

Definition 3.4.4 (Defeat) *An agent's ag_1 argument $a_1 \in AAFAS$ put forward in the context of a society S_t $defeats_{ag_1}$ other agent's $ag_2 \in AAFAS$ argument a_2 iff*
$attack(a_1, a_2) \wedge (val(ag_1, a_1) <^{S_t}_{ag_1} val(ag_1, a_2) \notin Valpref_{ag_1}) \wedge$
$(Role(ag_1) <^{S_t}_{Pow} Role(ag_2) \vee Role(ag_1) <^{S_t}_{Auth} Role(ag_2) \notin Dependency_{S_t})$

Therefore, we express that the argument a_1 $defeats_{ag_1}$ from the ag_1 point of view the argument a_2 as $defeats_{ag_1}(a_1, a_2)$ if a_1 attacks a_2, ag_1 prefers the value promoted by a_1 to the value promoted by a_2 and ag_2 does not have a power or authority relation over ag_1. Thus, based on Dung's acceptability semantics, we can define some acceptability concepts. Note that in them we compare arguments of different agents. However, since dependency relations are a partial order relations (reflexive, asymmetric and transitive), an agent has equal power, authorisation and dependency relations over itself ($ag \leq ag$ (reflexivity) $\rightarrow ag = ag$ (antisymmetry)) and, in case of comparing arguments of the same agent, the $AAFAS$ would be equivalent to an AVAF and the acceptability criteria of this AVAF would apply. Let $a_i, a_j, a_k \in A$ be the arguments of agents $ag_i, ag_j, ag_k \in Ag$ respectively and $a \in A$ the argument of a generic agent.

Definition 3.4.5 (Conflict-free) *A set of arguments $ARG \in A$ is $conflict-free_{ag_i}$ for an agent ag_i in the society S_t if*
$\nexists a_i, a_j \in ARG \ / \ (attacks(a_i, a_j) \vee attacks(a_j, a_i)) \wedge$
$((val(ag_i, a_i) <^{S_t}_{ag_i} val(ag_i, a_j) \notin Valpref_{ag_i}) \wedge$
$(val(ag_i, a_j) <^{S_t}_{ag_i} val(ag_i, a_i) \notin Valpref_{ag_i}) \wedge$
$(Role(ag_i) <^{S_t}_{Pow} Role(ag_j) \notin Dependency_{S_t}) \wedge$
$(Role(ag_j) <^{S_t}_{Pow} Role(ag_i) \notin Dependency_{S_t}) \wedge$
$(Role(ag_i) <^{S_t}_{Auth} Role(ag_j) \notin Dependency_{S_t}) \wedge$
$(Role(ag_j) <^{S_t}_{Auth} Role(ag_i) \notin Dependency_{S_t}))$.

That is, if there is no pair of arguments that attack each other, without a value preference relation or a dependency relation that invalidates the attack. Note that agent ag_i and ag_j can be the same, to consider the case of arguments put forward by the same agent.

Definition 3.4.6 (Acceptability) *An argument $a_i \in A$ is $acceptable_{ag_i}(a_i)$ in a society S_t wrt a set of arguments $ARG \in A$ iff $\forall a_j \in A \wedge defeats_{ag_i}(a_j, a_i) \rightarrow \exists a_k \in ARG \wedge defeats_{ag_i}(a_k, a_j)$.*

That is, if the argument is $defeated_{ag_i}$ by other argument of A, some argument of the subset ARG $defeats_{ag_i}$ this other argument.

Definition 3.4.7 (Admissibility) *A conflict-free set of arguments $ARG \in A$ is admissible for an agent ag_i iff $\forall a \in ARG \rightarrow acceptable_{ag_i}(a)$.*

Definition 3.4.8 (Preferred Extension) *A set of arguments $ARG \in A$ is a $preferred-extension_{ag_i}$ for an agent ag_i if it is a maximal (wrt set inclusion) $admissible_{ag_i}$ subset of A.*

Then, for any $AAFAS = < Ag, Rl, D, G, N, A, R, V, Role, Dependency_{S_t}, Group, Values, val, Valpref_{ag_i} >$ there is a corresponding $AFAS = <A, R, S_t >$, where $R = defeats_{ag_i}$. Thus, each attack relation of $AFAS$ has a corresponding agent specific $defeat_{ag_i}$ relation in $AAFAS$. These properties will be illustrated in the mWater study case that will be presented in Chapter 5. The following section instantiates the proposed abstract argumentation framework for agent societies. Thus, it provides a formal specification for the framework and defines the structure and property relations of its elements.

3.5 Case-based Argumentation Framework for Agent Societies

Following our case-based computational representation of arguments, we have designed a formal argumentation framework (AF) as an instantiation of Dung's AF [Dung, 1995]. The main advantages that our framework contributes over other existent AFs deal with the requirements suggested in Section 3.2.2. These advantages are: 1) the ability to represent social information in arguments; 2) the possibility of automatically managing arguments in agent societies; 3) the improvement of the agents' argumentation skills; and 4) the easy interoperability with other frameworks that follow the argument and data interchange web standards. According to Prakken [Prakken and Sartor, 1996], the elements that characterise an AF are: the notion of argument used in the framework, the logical language that represents argumentation concepts, the concept of conflict between arguments, the notion of defeat between arguments and the acceptability status of arguments. Next, these elements are specified by using the knowledge resources defined in this section and the ArgCBROnto ontology.

3.5.1 The Notion of Argument: Case-Based Arguments

We have adopted the Argument Interchange Format (AIF) [Willmott et al., 2006] view of arguments as a set of interlinked premiss-illative-conclusion sequences. The notion of argument is determined by our KI case-based framework to represent arguments. In

our framework agents can generate arguments from previous cases (domain-cases and argument-cases), from argumentation schemes or from both. 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 between agents is not the same. Thus, arguments that agents interchange are defined as tuples of the form:

Definition 3.5.1 (Argument) *$Arg = \{\phi, v, < S >\}$, where ϕ is the conclusion of the argument, v is the value that the agent wants to promote with it and $< S >$ is a set of elements that support the argument (support set).*

In the ArgCBROnto ontology, in addition of the above elements (see Figure 3.8), arguments have a unique identifier *ID* (which ontological definition was provided in Section 3.3.1):

$Argument \sqsubseteq Thing$

$SupportSet \sqsubseteq Thing$

$Argument \sqsubseteq \forall hasConclusion.Conclusion$

$Argument \sqsubseteq \exists hasConclusion.Conclusion$

$Argument \sqsubseteq = 1 hasConclusion$

$Argument \sqsubseteq \forall promotesValue.Value$

$Argument \sqsubseteq \exists promotesValue.Value$

$Argument \sqsubseteq = 1 promotesValue$

$Argument \sqsubseteq \forall hasSupportSet.SupportSet$

$Argument \sqsubseteq \exists hasSupportSet.SupportSet$

$Argument \sqsubseteq = 1 hasSupportSet$

This support set can consist of different elements, depending on the argument purpose. On one hand, if the argument provides a potential solution for a problem, the support set is the set of features (*premises*) that describe the problem to solve and optionally,

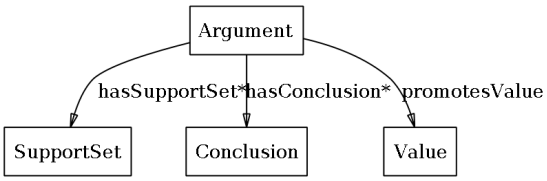

Figure 3.8: ArgCBROnto Argument

any knowledge resource used by the proponent to generate the argument (*domain-cases*, *argument-cases* or *argumentation schemes*). On the other hand, if the argument attacks the argument of an opponent, the support set can also include any of the allowed attacks in our framework (*critical questions (presumptions and exceptions)*, *distinguishing premises* or *counter-examples*). Then, the support set consists of the following tuple of sets of support elements [3]:

Definition 3.5.2 (Support Set) $S = <\{premises\}, \{domainCases\}, \{argumentCases\}, \{argumentationSchemes\}, \{criticalQuestions\}, \{distinguishingPremises\}, \{counterExamples\} >$

In the ArgCBROnto ontology, the elements of the support set are represented with the following properties (see Figure 3.9):

$SupportSet \sqsubseteq \forall hasPremise.Premise$

$SupportSet \sqsubseteq \forall hasDomainCase.DomainCase$

$SupportSet \sqsubseteq \forall hasArgumentCase.ArgumentCase$

$SupportSet \sqsubseteq \forall hasArgumentationScheme.ArgumentationScheme$

$SupportSet \sqsubseteq \forall hasPresumption.Premise$

$SupportSet \sqsubseteq \forall hasException.Premise$

$SupportSet \sqsubseteq \forall hasDistinguishingPremise.Premise$

$SupportSet \sqsubseteq \forall hasCounterExample.Case$

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

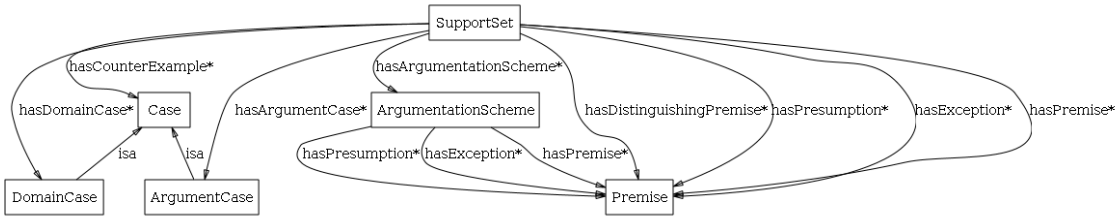

Figure 3.9: ArgCBROnto SupportSet

3.5.2 The Logical Language

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

The vocabulary of cases and schemes is defined by using the ArgCBROnto ontology previously presented. We have selected the Ontology Web Language OWL-DL[4] as the formal logics to represent the vocabulary of cases. This variant is based on Description Logics (DL) and guarantees computational completeness and decidability. Thus, it allows automatic description logic reasoning over argument-cases and domain-cases. In addition, it facilitates the interoperability with other systems. In Sections 3.3 and 3.5.1, we have provided a partial view of the ontology for the AF proposed[5].

3.5.3 The Concept of Conflict between arguments

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

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

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

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

Definition 3.5.3 (Rebut) *Arg_1 rebuts Arg_2 iff*
$\phi_1 = \sim\phi_2$ *and* $\{Premises\}_1 \supseteq \{Premises\}_2$

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

Definition 3.5.4 (Undercut) *Arg_1 undercuts Arg_2 if*
$1)\phi_1 = \sim conc(as_k)/$
$\exists cq \in \{CriticalQuestions\}_1 \wedge \exists as_k \in \{ArgumentationSchemes\}_2 \wedge$
$cq \Rightarrow \sim conc(as_k),$ *or*
$2)\phi_1 = dp/$
$(\exists dp \in \{DistinguishingPremises\}_1 \wedge \exists pre_k \in \{Premises\}_2 \wedge dp = \sim pre_k) \vee$
$(dp \notin \{Premises\}_2),$ *or*
$3)\phi_1 = ce/$
$(\exists ce \in \{CounterExamples\}_1 \wedge \exists dc_k \in \{DomainCases\}_2$
$\wedge\, conc(ce) = \sim conc(dc_k)) \vee$
$(\exists ce \in \{CounterExamples\}_1 \wedge$
$\exists ac_k \in \{ArgumentCases\}_2 \wedge conc(ce) = \sim conc(ac_k))$

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

3.5.4 The Notion of Defeat between arguments

Once possible conflicts between arguments have been defined, the next step in the formal specification of an AF is to define the defeat relation between a pair of arguments. This comparison must not be misunderstood as a strategical function to determine with which argument an argumentation dialogue can be won [Prakken and Sartor, 1996]. A function like this must also consider other factors, such as other arguments put forward in the dialogue or agents' profiles. Therefore, it only tells us something about the relation between two arguments. Hence, the relation of defeat between two arguments is defined in our AF as follows.

Let $Arg_1 = \{\phi_1, value_1, < S_1 >\}$ posed by agent ag_1 and $Arg_2 = \{\phi_2, value_2, < S_2 >\}$ posed by agent ag_2 be two conflicting arguments and $Valpref_{ag_i} \subseteq VxV$, defines an irreflexive, antisymmetric and transitive relation $<^{S_t}_{ag_i}$ over the agent's ag_i values in the society S_t. Then:

Definition 3.5.5 (Defeat) *Arg_1 defeats Arg_2 if ((rebuts(Arg_1,Arg_2) $\wedge \sim$ undercut(Arg_2,Arg_1)) $\vee$ undercuts(Arg_1,Arg_2)) $\wedge$ ($value_1$ $<^{S_t}_{ag_1}$ $value_2 \notin Valpref_{ag_1}$) $\wedge$ ($Role(ag_1) <^{S_t}_{Pow} Role(ag_2) \notin Dependency_{S_t} \wedge Role(ag_1) <^{S_t}_{Auth} Role(ag_2) \notin Dependency_{S_t}$)*

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

Definition 3.5.6 (Strict Defeat) *Arg_1 strictly defeats Arg_2 if Arg_1 defeats Arg_2 and Arg_2 does not defeat Arg_1*

3.5.5 The Acceptability State of arguments

The acceptability status of arguments determines their status on the basis of their interaction. Only comparing pairs of arguments is not enough to decide if their conclusions are acceptable, since defeating arguments can also be defeated by other arguments.

Taking into account the underlying domain theory of a dialectical system, arguments can be considered *acceptable*, *unacceptable* and *undecided* [Dung, 1995]. However, the acquisition of new information in further steps of the dialogue could change the acceptability status of arguments.

Therefore, to decide the acceptability status of arguments a proof theory that takes into account the dialogical nature of the argumentation process is necessary. To evaluate the acceptability of arguments by using a dialogue game is a common approach [Heras et al., 2009d]. Dialogue games are interactions between two or more players, where each one moves by posing statements in accordance with a set or predefined rules [McBurney and Parsons, 2002a]. In our AF, the acceptability status of arguments is decided by using a dialogue game and storing in the argument-case associated to each argument its acceptability status when the dialogue ends. The definition of this dialogue game is provided in Chapter 4.

3.6 Reasoning Process

This section presents the reasoning process that agents of our AF follow to generate and select their positions and arguments and to evaluate them in view of the other agents' positions and arguments. With this process, agents are able to automatically argue and learn from the argumentation experience. Here, we first define some concepts that will be used in the following sections.

Definition 3.6.1 (Function Value) *The function value is defined as* $value_k(x) : C \times F \rightarrow V$ *and returns for a case* k *(from a set of cases* C*) the value of the feature* x *(from a set of features* F_k *of the case* k *and a set of values* V*).*

Definition 3.6.2 (Match) *A match between two sets of features* $i, j \in F$ *is defined as:* $match(i, j) : F \times F \rightarrow true$ *iff* $F_i \cap F_j \neq \emptyset$ *and* $\forall f \in F_i \cap F_j,\ val_i(f) = val_j(f)$*. Hence, two cases match if the features that describe them match.*

Definition 3.6.3 (Subsumption) *A case* c_l *subsumes other case* c_m *(from a set of cases* C*):* $subsumes(c_l, c_m) : C \times C \rightarrow true$ *iff* $match(c_l, c_m)$ *and* $\forall f_m \in c_m, \exists f_l \in c_l / val_{c_l}(f_l) = val_{c_m}(f_m)$*.*

Therefore, we also describe problems as cases without solution and assume that a match between the problem to solve and a stored case means that the latter has some features

of the problem and with the same values[6]. A total match between a problem and a case or between two cases means that both cases have the same features and with the same values.

Definition 3.6.4 (Counter-Example) *A counter-example for a case is a previous domain-case or an argument-case that was deemed acceptable, where the problem description of the counter-example matches the current problem to solve and also subsumes the problem description of the case, but proposing a different solution.*

Definition 3.6.5 (Distinguishing Premise) *A distinguishing premise $x \in P$ with respect to a problem P between two cases $c_1, c_2 \in C$ is defined as: $\exists x \in c_1 / \exists h \in c_2 \wedge x = h \wedge value_{c_1}(x) \neq value_{c_2}(x)$ or else, $x \notin c_2$, where $x, h \in F$ and $c_1, c_2 \in C$.*

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

3.6.1 Position Management

In the first step to reach an agreement about the best solution for a problem to solve, an agent can generate its individual position, which represents the best solution for the problem from the agent's point of view. At this level of abstraction, we assume that this is a generic problem of any type (e.g. a classification, a prediction, etc.) that could be described with a set of features. This is not a compulsory step for agents to engage in the argumentation process to reach the agreement, since some agents could argue about positions of others' without having necessarily generated their own's. Then, with the set of generated positions agents generate associated argument-cases to support them. After that, they can use their case-bases of argument-cases to select the best position to propose. Finally, agents evaluate their positions in view of the others' positions and their previous experience.

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

3.6.1.1 Position Generation

In our AF, agents have several ways to generate positions, depending on their design or even on strategical considerations. Thus, an agent could follow different mechanisms to generate positions:

1. **From the Problem Description and Domain-Cases:** This option would be followed by agents that rely more on their experiences. The agent retrieves from the domain case-base those cases that match with the specification of the current problem. With the solutions that were applied in these cases, the agent generates a potential solution for the problem at hand, which represents its position with respect to the problem. Note that the set of retrieved cases could provide different solutions for the same problem. For instance, assuming that the agent has to provide a solution for the problem $p = (f_1, f_2, f_3)$ that partially matches with two cases $c = (f_1, f_2, s_c)$ and $d = (f_2, f_3, s_d)$ of its case-base with solutions s_c and s_d respectively. Then, positions $pos_1 = (f_1, f_2, s_c)$ and $pos_2 = (f_2, f_3, s_d)$ can be created. Thus, in this situation we have that $value_p(f_1) = value_c(f_1), value_p(f_2) = value_c(f_2) = value_d(f_2)$ and $value_p(f_3) = value_d(f_3)$.

2. **From the Problem Description and Argumentation Schemes:** This option would be followed by agents that rely more on its pre-defined argumentation schemes. Then, an agent can generate its position as the conclusion drawn by using the problem description to match (totally or partially) the premises of a scheme. For instance, assuming that there is an scheme $AS1 = (premise_1, premise_2, conclusion)$ that partially matches the problem p (with $premise_1$ matching f_1 and $premise_2$ matching f_2), the position $pos_3 = (f_1, f_2, s_AS1)$, where $s_AS1 = conclusion$ could be created. Also, several positions could be generated if the description matches the premises of several schemes.

3. **From the Problem Description, Cases and Argumentation Schemes:** This option would be selected by agents that prefer to exploit all their resources and follow an hybrid generation policy. The agent retrieves from its domain case-base the cases that match the current problem. Then, it can extend the problem description by adding the set of attributes of the retrieved cases that are consistent with this description (do not appear in the problem description) and with the set of retrieved cases (have the same value in all retrieved cases that they appear). For instance, assume that the agent has to provide a solution for the problem p

that matches again with case c and d, but this time, the cases have two extra features f_5 and f_6 ($c = (f_1, f_2, f_5, f_6, s_c)$ and $d = (f_2, f_3, f_5, f_6, s_d)$). Also, both cases have the same value for f_5, but different values for f_6 ($value_c(f_5) = value_d(f_5)$ but $value_c(f_6) \neq value_d(f_6)$). Thus, features f_6 in cases c and d are considered inconsistent and only feature f_5 would be added to the extended problem description ($p' = (f_1, f_2, f_3, f_5)$). Finally, from this new problem description and the argumentation schemes the agent can generate its position (or positions), as explained above. This method for generating positions follows the idea of broadening the space of possible solutions by considering features that, despite not being specified in the current problem, have been observed in similar problems in the past. Note that only the cases that match the problem description are retrieved and hence, the space of potential positions could not be extended if we only consider positions generated from cases. However, the extended problem description could add a new feature that makes the description to match an argumentation scheme that was not considered before.

Algorithm 1 shows the pseudocode of the process to generate positions from domain-cases, argumentation schemes or both of them. In the algorithm, *DomainCasesCB* represents the case-base of domain-cases and *ArgumentationSchemesOnt* represents the ontology of argumentation schemes. If the generation method to follow is "D", the algorithm generates positions from the case-base of domain-cases. In case that the method is "S", the algorithm generates positions from the ontology of argumentation schemes. Similarly, if the method to follow is "M", the algorithm generates positions from both the case-base of domain-cases and the ontology of argumentation schemes.

Also, *computeSimilarity* is a domain-dependent function that computes the similarity between a current problem and the description of the domain-cases or argumentation schemes stored in the knowledge resources of the system. This similarity degree is stored for each potential position to solve the problem (when the similarity degree exceeds a pre-defined threshold). *generateSolutions* is a domain-dependent function that generates potential solutions for the problem to solve from the solutions of the domain-cases that are deemed similar to the current problem or the conclusions of the similar argumentation schemes. *addPosition* is a function that adds a new position to the list of potential solutions for the problem to solve. *aggregateDescriptions* is a function that adds to the problem description the extra consistent features that are found in the problem description of the similar domain-cases. Finally, *addGenerationMethod* is a function that forces the algorithm to execute a specific generation method.

3.6.1.2 Argument-case Generation

For each position generated, the agent creates an argument-case that represents this position (recalling, a potential solutions for the problem) and its support. Thus, it fills the argument-case structure with the available information for each element. Note that, depending on the actual problem to solve and the type of dialogue, the proponent agent could not know some data about its opponent (e.g. in negotiation dialogues agents are usually unwilling to share information about its values or preferences with other agents). The effects of this knowledge uncertainty will be discussed in Section 3.7.

To show all possibilities that offer the argument-case structure, in this section we assume a two-party dialogue between agents of the same group, as pointed out in Section 3.3. Moreover, we assume that the agents have previous knowledge about each other, so the proponent can fill in the social context information of the opponent. However, in this new argument-case, the acceptability status and attacks features in the solution part of the argument-case cannot still be filled in. Note that, actually, positions are subsets of elements of argument-cases that include the features of the domain context and the conclusion.

3.6.1.3 Position Selection

With the set of potential positions, the agent has to decide the one it will propose first. Even when the agent has generated only one position, this step allows it to check if this position is worth to defend and until which extent. The first step for this selection is to order the positions in subsets, taking into account the value that promotes each position. Thus, the agent will assign each set a *Suitability Level (SL)*. Positions that promote the value most preferred by the agent will be labelled with suitability level 1, positions that promote the second most preferred value will be labelled with suitability level 2 and so on. After that, positions will be ordered in each level by its *Similarity Degree (SimD)* with the problem to solve.

Finally, the agent will consider the argumentation knowledge stored in its argument-cases case-base. Therefore, the agent compares the argument-case of each position generated with its case-base of argument-cases and retrieves in the set arg those which problem description match the problem description of the current argument-case. To do that, the agent extends the position description with the current social context and

retrieves the set of similar argument-cases. This set represents the previous experience of the agent in similar argumentation processes, taking into account the social context. Then, the agent can assign each position a (*Suitability Factor* (SF)) from the argumentation point of view.

Actually, what the agent does is to decide which argument-case (and thus, which position) is most suitable in view of its past experience. We consider the parameters shown in the following formulas as criteria for making such decision. In the formulas, $argC$ is the number of argument-cases in arg with the same conclusion than the current argument-case, $argAccC$ are those in $argC$ that were deemed acceptable, $argAccCAtt$ are those in $argAccC$ that were attacked, $minAtt$ and $maxAtt$ are the minimum and maximum number of attacks received by any position generated, $minS$ and $maxS$ are the minimum and maximum number of steps from any retrieved argument-case to the last node of its dialogue graph and $minKr$ and $maxKr$ are the minimum and maximum number of knowledge resources used to generate any position.

- **Persuasiveness Degree (PD):** is a value that represents the expected persuasive power of a position by checking how persuasive an argument-case with the same problem description and conclusion that the position associated argument-case was in the past. To compute this degree, the number $argAccC$ of argument-cases that were deemed acceptable out of the total number of argument-cases $argC$ with the same problem description and conclusion retrieved is calculated:

$$PD = \begin{cases} 0, & if\ argC = \emptyset \\ \dfrac{argAccC}{argC}, & otherwise \end{cases} \tag{3.1}$$

  with $argAccC$, $argC \in N$ and $PD \in [0, 1]$, from less to more persuasive power. Note that we do not decrease de persuasiveness degree of a position if positions with the same problem description and different conclusions are found in the argument-base, since this difference does not necessarily implies that the current position is wrong or less persuasive. In fact, there are many possible reasons for having different conclusions for the same problem description (e.g. different background knowledge, different reasoning algorithms to generate positions or even a domain admitting several solutions for the same problem).

- **Support Degree (SD):** is a value that provides an estimation of the probability that the conclusion of the current argument-case was acceptable at the end of the

dialogue. It is based on the number of argument cases $argAccC$ with the same problem description and conclusion that where deemed acceptable out of the total number of argument-cases arg retrieved.

$$SD = \begin{cases} 0, & if\ arg = \emptyset \\ \dfrac{argAccC}{arg}, & otherwise \end{cases} \tag{3.2}$$

with $argAccC, arg \in N$ and $SD \in [0, 1]$ from less to more support degree.

- **Risk Degree (RD):** is a value that estimates the risk for a position to be attacked in view of the attacks received for a position(s) with the same problem description and conclusion in the past. It is based on the number of argument cases $argAccCAtt$ that were attacked out of the total number of $argAccC$ argument cases with the same problem description and conclusion retrieved that were deemed acceptable.

$$RD = \begin{cases} 0, & if\ argC = \emptyset \\ \dfrac{argAccCAtt}{argAccC}, & otherwise \end{cases} \tag{3.3}$$

with $argAccCAtt$, $argC \in N$ and $RD \in [0, 1]$, from less to more risk of attack.

- **Attack degree (AD):** is a value that provides an estimation of the number of attacks att received by a similar position(s) in the past. To compute this degree, the set of arguments with the same problem description that were deemed acceptable is retrieved. Then, this set is separated in several subsets, one for each different conclusion. The sets whose conclusion match with the conclusions of the positions to assess are considered, while the other sets are discarded. Thus, we have a set of argument-cases for each different position we want to evaluate. For each argument-case in each set, the number of attacks received is computed (the number of critical questions, distinguishing premises and counter-examples received). Then, for each set of argument-cases, the average number of attacks received is computed. The attack degree of each position is calculated by a linear transformation:

$$AD = \begin{cases} 0, & if\ maxAtt = minAtt \\ \dfrac{att - minAtt}{maxAtt - minAtt}, & otherwise \end{cases} \tag{3.4}$$

with $minAtt$, $maxAtt$, $att \in N$ and $AD \in [0, 1]$ from less to more degree of

attack.

- **Efficiency degree (ED):** is a value that provides an estimation of the number of steps that took to reach an agreement posing a similar position(s) in the past. It is based on the depth n from the node representing the argument-case of the similar position to the node representing the conclusion in the dialogue graphs associated to the similar argument-cases retrieved. To compute this degree, the same process to create the subsets of argument-cases than in the above degree is performed. Then, for each argument-case in each subset, the number of dialogue steps from the node that represents this argument-case to the end of dialogue is computed. Also, the average number of steps per subset is calculated. Finally, the efficiency degree of each position is calculated by a linear transformation:

$$ED = \begin{cases} 0, & if\ maxS = minS \\ 1 - \dfrac{n - minS}{maxS - minS}, & otherwise \end{cases} \tag{3.5}$$

  with $minS$, $maxS$, $n \in N$ and $ED \in [0, 1]$ from less to more efficiency.

- **Explanatory Power (EP):** is a value that represents the number of pieces of information each position covers. It is based on the number kr of knowledge resources were used to generate each position. To compute this number, the same process to create the subsets of argument-cases than in the above degrees is performed. Then, for each argument-case in each set, the number of knowledge resources in the justification part is computed (the number of domain-cases, argument-cases and argumentation schemes). Then, for each set of argument-cases, the average number of knowledge resources used is computed. The explanatory power of each position is calculated by a linear transformation:

$$EP = \begin{cases} 0, & if\ maxKr = minKr \\ \dfrac{kr - minKr}{maxKr - minKr}, & otherwise \end{cases} \tag{3.6}$$

  with $minKr$, $maxKr$, $kr \in N$ and $EP \in [0, 1]$ from less to more explanatory power.

Finally, the suitability factor of a new argument-case and its associated position is

computed by the formula:

$$\begin{aligned} SF =&((w_{PD} * PD + w_{SD} * SD + w_{RD} * (1 - RD) \\ &+ w_{AD} * (1 - AD) + w_{ED} * ED + w_{EP} * EP)) \end{aligned} \tag{3.7}$$

where $w_i \in [0, 1], \sum w_i = 1$ are weight values that allow the agent to give more or less importance to each decision criteria. Finally, positions are ordered from more to less suitability by following the equation:

$$Suitability = w_{SimD} * SimD + w_{SF} * SF \tag{3.8}$$

where $w_i \in [0, 1], \sum w_i = 1$ are weight values that allow the agent to give more or less importance to the similarity degree or the support factor. Finally, the most suitable position of suitability level 1 is selected as the one that the proponent agent is going to propose and defend first. Then, we assume that agents follow their value preference criteria when they select the positions to propose.

Algorithm 2 shows the pseudocode of the algorithm that implements the generation of positions, the generation of the associated argument-cases and the selection of positions. In the algorithm, the function *generatePositions* generates the n first positions by using the Algorithm 1; *generateArgumentCase* is a function that generates for each position its associated argument-case; *retrieveSimilarityDegree* is a function that retrieves the similarity degree of each position with regard to the problem to solve; *selectPosition* is a domain-dependent function that orders the set of positions from more to less suitable with respect to some domain-dependent criteria; and *mostSuitable* is a domain-dependent function that returns the most suitable position to solve the problem.

Also, *computeSF*, as shown in Algorithm 3, is a function that computes the support factor for each position by means of its associated argument-case. In this algorithm, the function *retrieveSameProblem* retrieves from the case-base of argument-cases those that have the same problem description than the current argument-case; *retrieveSameConclusion* retrieves from the case-base of argument-cases those that have the same problem description and conclusion than the current argument-case; *retrieveAccepted* retrieves from the case-base of argument-cases those that have the same problem description and conclusion than the current argument-case and were deemed acceptable; *retrieveAcceptedAttacked* retrieves from the case-base of argument-cases those that have the same problem description and conclusion than the current argument-case, were deemed acceptable and were attacked; *computeNumberOfAttacks* computes the number of at-

tacks received by an argument-case; *computeNumberOfSteps* computes the number of steps from an argument-case to the node that represents the final conclusion in its associated dialogue-graph; and *computeNumberOfKR* computes the number of knowledge resources used to generate an argument-case.

3.6.1.4 Position Evaluation

In addition to generate its position, a proponent agent can evaluate it in view of the positions that other opponent agents put forward in the dialogue. This step is only performed if the proponent has knowledge about the opponents' positions. Otherwise, it could just try to defend its position when it receives attacks from them (see Section 3.6.2). The first step to evaluate an agent's position is to check if it is consistent with the opponents' positions. This is performed by means of a *similarity function* that is domain-dependent. Generally, we assume that a position is consistent with other position if they are the same (they totally match)[7]. As the original description of the problem is the same for every agent in the dialogue, the proponent only needs to check if the opponent's position matches with one of the positions that it generated. Then:

- If the opponent's position matches the proponent's position, the opponent is considered as a supporter and both agents agree in the solution for the problem. Thus, no attacks are necessary, but the proponent can generate an argument to defend its positions if the position is attacked by other agent.

- If the opponent's position is in the set of positions generated by the proponent, but not ranked as the most suitable position, the opponent could either accept it and change its mind or, on the contrary, try to generate an attack argument to the opponent's position. In this case, we assume that the proponent would accept the position of the opponent if the latter has a power or authorisation[8] relation over the proponent and attack the opponent's position otherwise.

- If the opponent's position is not in the set of positions generated by the proponent and the opponent does not have a power or an authorisation[5] relation over the

[7]Broaden notions of consistency, such as one position being part or a more general position (e.g. an action that is part of a course of action proposed to solve a problem), can be considered in specific domains.

[8]We assume that these power and authorisation dependency relations hold for the issue under discussion.

proponent, the proponent can generate an argument to attack the opponent's position. Otherwise, it must accept the opponent's position.

The decision mechanisms that an agent can use to perform a specific action in each case will be explained in Chapter 4. Next section explains the type of arguments that agents can generate and how these arguments are selected and evaluated.

Algorithm 4 shows the pseudocode of the position evaluation process. In the algorithm, the function *checkDependencyRelation* checks the dependency relation between the proponent and the opponent. As explained above, if the opponent's position is in the list of potential positions of the proponent but not ranked first, the proponent can use the function *decideAttack* to decide if it would attack the incoming position or just change its preferences. Also, *askForSupport* is a function that an agent can use to ask other agent to support its position.

3.6.2 Argument Management

Depending on their purposes, agents can generate different types of arguments, select the best argument to put forward in view of a specific situation and evaluate the arguments of other agents. In our AF, arguments that agents interchange are tuples of the form of Definition 3.5.1. In addition, the argument management process depends on the type of information that an agent receives from other agent. Here, we assume that this type can be identified from the type and content of the locutions that agents receive from other agents.

3.6.2.1 Argument Generation

Agents generate arguments when they are asked to provide evidence to support a position (*support arguments*) or when they want to attack others' positions or arguments[9] (*attack arguments*).

The fist case happens because, by default, agents are not committed to show evidences to justify their positions. Therefore, an opponent has to ask a proponent for an argument that justifies its position before attacking it. Then, if the proponent is willing to offer support evidences, it can generate a support argument which support set is the

[9]For the time being, we do not consider arguments that agents could generate to support others' positions and arguments.

set of features (premises) that describe the problem and match the knowledge resources that it has used to generate and select its position and any of these resources (domain-cases, argument-cases, argumentation schemes or partial views of them). Note that the set of premises could be a subset of the features that describe the problem to solve (e.g. when a position has been generated from a domain-case that has a subset of features of the problem in addition to other different features).

The second case happens when the proponent of a position generates an argument to justify it and an opponent wants to attack the position or more generally, when an opponent wants to attack the argument of a proponent.

Algorithm 5 shows the pseudocode of the argument generation process. In the algorithm, the function *evaluateIncomingRequest* evaluates the type of the incoming request received. If the agent's position has been asked for support, the agent can decide to generate it by using the domain-dependent function *decideSupport*. If the agent receives an attack, it must evaluate its current argument in view of the attacking argument by using the function *evaluateArgument*, explained in Section 3.6.2.4.

The attack arguments that the opponent can generate depend on the elements of the support set of the argument of the proponent:

- If the justification for the conclusion of the argument consists of a set of premises, the opponent can generate an attack argument with a distinguishing premise that it knows. It can do it, for instance, if it is in a privileged situation and knows extra information about the problem or if it is implicit in a case that it used to generate its own position, which matches the problem specification. In the latter, the opponent could generate an attack argument with this case as counter-example.

- If the justification has an argumentation scheme and if the opponent knows the scheme (e.g. the systems shares an ontology of argumentation schemes with all agents), it can generate an argument that invalidates the proponent's argument proposing a critical question associated to the scheme. The specific process depends on the ontology and the application domain and is not specified here. On the contrary, the opponent can temporally store the scheme to decide later if it should be added to its argumentation schemes ontology.

- If the justification has a domain-case or an argument-case, then the opponent can check its case-bases of domain-cases and argument-cases and try to find counter-examples to generate an attack argument with them. Alternatively, it can also

try to generate an attack argument with a distinguishing premise from its own known premises and cases that invalidates the proponent's justification.

However, even if the proponent was not willing to show any evidence (there is no support set) the opponent can still attack the proponent's position by showing an own domain-case or argument-case that acts as a counter-example for the position. The same happens when the justification of the argument that the proponent offers is an argumentation scheme that the opponent does not know. In this case, the opponent can also generate an attack argument with a counter-example for the conclusion of the proponent's argument.

Algorithm 6 presents the pseudocode of the attack generation process. In the algorithm, the function *checkSupportSet* checks the elements of the support set of the incoming argument. With the function *selectElementToAttack*, the agent selects which element(s) of the support set it wants to attack. By means of *genetateDPAttack* the agent tries to attack the incoming argument with a distinguishing premise or a counter-example with a distinguishing premise. The function *generateASAttack* tries to attack the incoming argument with a critical question of the argumentation scheme that supports the incoming argument. Also, *storeAS* is a function that agents can use to store an unknown scheme and decide later if it will be added to the ontology of argumentation schemes. With the function *generateCounterExample* the agent tries to generate a counter-example from its case-bases of domain-cases or argument-cases and with *generateCEAttack* the agent tries to attack the incoming argument with the counter-example.

3.6.2.2 Argument-case Generation

As explained in the above section, an agent can generate support arguments to support its position or attack arguments to attack others' position and arguments. In the case of support arguments, the argument shows information of the argument-case associated to the agent's position. Depending on the specific implementation, the agent can generate a support argument that shows all the information that the agent has used to generate the position (stored in the associated argument-case) or only partial information. However, in any case the agent has generated yet the argument-case that supports its position, as explained in Section 3.6.1.2 and generating a new argument-case associated to the support argument is not necessary.

For attack arguments, a new argument-case must be generated to store the information about this step of the argumentation process. This argument-case can be generated from the context of the current argumentation process by following a process like the one explained in Section 3.6.1.2. However, in this case the conclusion of the argument-case is a rebut or an undercut to the conclusion of the argument that the agent wants to attack and not a solution for the problem at hand.

3.6.2.3 Argument Selection

Depending on the content of their knowledge resources, agents can generate different arguments for supporting their positions or for attacking positions and arguments of other agents. An agent could have generated several arguments by using different knowledge resources of the same type or different combinations between them. Thus, the agent has to select the best argument to put forward from the set of potential candidate arguments.

In the case of argument-cases associated to support arguments, we assume that arguments that provide more justifications for a position are more persuasive and should be proposed first. However, agents can follow different domain-dependent strategies to select the best support argument to put forward. Note that the criteria defined in Section 3.6.1.3 cannot be used since all possible support arguments generated are represented by the same argument-case and have the same conclusion (the position proposed by the agent to solve the problem). We do not generate a new argument-case for each possible support argument to avoid the overload of the case-base of argument-cases.

In the case of attack arguments, the agent can use the information gained from previous argumentation processes to decide which argument would have more *Suitability degree*, just as done for positions selection in Section 3.6.1.3. In case of a draw in the suitability degrees of the most suitable arguments, the agent has to decide which argument is going to put forward. Many times, this decision depends on the implementation of the agent and the dialogue strategy that it follows. However, in our AF at least there is a reflexive and transitive pre-order relation $<_p$ among the persuasive power of the knowledge resources used to generate an argument: $premises <_p distinguishing - premises <_p argumentation - schemes <_p critical - questions <_p domain - cases <_p argument - cases$. Accepted argument-cases are the most persuasive knowledge resource to show as a justification for an argument, since they store the the maximum quantity of information about past arguments and argumentation pro-

cesses (domain context, social context, if it was attacked and still remained accepted, etc.). Domain-cases are also very persuasive, since they store the final solution applied for a problem, but they do not provide information about the argumentation process and the social context. Like them, critical questions and argumentation schemes do not provide these information and besides, they store general knowledge about argumentation, but do not show any real past experience. Also, critical questions can invalidate the conclusion drawn from argumentation schemes and hence, they have more persuasive power. Finally, the premises that describe the problem to solve are known by any agent in the argumentation process and have the lowest persuasive power. In the case of attack arguments, distinguishing premises are more persuasive than description premises, since they provide information that. Therefore, argument-cases with at least one argument-case in the justification part would be preferred to others and so on. If even in these case the draw persists, a random choice could be made.

3.6.2.4 Argument Evaluation

When agents receive arguments from other agents, they have to evaluate them in view of the current problem to solve and their knowledge resources. Then, a proponent agent can decide if an opponent's argument conflicts with its argument and hence, its argument is deemed acceptable, non-acceptable or undecided (it cannot make a decision over it) from its point of view. This decision can be determined by many factors. For instance, in systems where dependency relations must be always observed, subordinates could always have to accept arguments from superiors. Here, we assume a less restrictive domain and define some concepts that agents of our AF use to evaluate arguments.

In our AF, we define the two typical types of attacks between arguments in argumentation theory (*rebuts* and *undercuts*) and the notion of *defeat* as explained in Sections 3.5.3 and 3.5.4. Finally, if the proponent considers that its argument defeats the opponent's argument, it can try to generate a new argument to attack the opponent's, which would change the preliminary acceptability status of the opponent's argument to non-accepted. Then, the opponent would evaluate the proponent's argument and try to rebut or undercut the attack. On the contrary, if the proponent's argument cannot defeat the opponent's, it can still try to withdraw it and send the opponent's a different argument to support its position. In this case, the proponent's argument acceptability status would preliminary change to undecided. The defeat relation must

not be misunderstood as a strategical function to determine with which argument an argumentation process can be won. A function like this is domain-dependent and must also consider other factors, such as other arguments put forward in the dialogue, social contexts, etc. Therefore, agents only use it to make partial decisions about the relation between two arguments. The final acceptability status of arguments would be set at the end of the dialogue when the agreement process is reached (or the dialogue just ended without agreement).

Algorithm 7 shows the pseudocode of the argument evaluation process. As pointed out above, if the proponent argument defeats the opponent argument, it can try to attack it by using the function *generateAttack*, shown in Algorithm 6. With the function *generateNewSupport* the agent can try to generate a new support for its position (since it cannot defeat the opponent argument). However, if no new support can be generated, the agent has to withdraw its position from the dialogue by using the function *withdraw*.

3.7 Conclusions

In this chapter we propose a computational framework for argumentation in open MAS where agents can form societies and have values and preference relations over them. First, we provide an abstract definition of the framework, a an extension of abstract value-based argumentation frameworks. Then, the framework is instantiated by providing a concrete definition of the notion of argument, the logical language used to characterise arguments, the conflict relation over arguments, the notion of defeat and the acceptability status of arguments. A generic KI case-based structure for computational argument representation has been presented. This structure was inspired by the standard for argument interchange on the web (AIF) and hence, an argumentation system based on it can interact with other systems that comply with the standard. Moreover, elements of cases are specified by using an ontologic case representation language, the ArgCBROnto ontology. Although agents in open MAS are heterogeneous, by only sharing this ontology they can *understand* the arguments interchanged in the system. An example of the translation between some concepts of ArgCBROnto and their equivalent concepts in AIF (in the version reported in [Rowe and Reed, 2008]) is shown in table 3.3.

AIF represents actual arguments as graphs with interlinked nodes that stand for the different concepts of the AIF ontology [Willmott et al., 2006] [Rowe and Reed, 2008].

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

Table 3.3: Correspondence between ArgCBROnto and AIF concepts

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

Finally, this work also presents the automatic reasoning process to generate, select and evaluate positions and arguments, providing the pseudocode of the algorithms used in each of these processes.

For simplicity purposes, we have assumed in this chapter that a proponent agent addresses its arguments to an opponent of its same group, having complete knowledge of the 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, probably knows the preferences of its group but, in a real open MAS, it is

unlikely to know the opponent's value preferences. However, the proponent could know the value preferences of the opponent's group or have some previous knowledge about the value preferences of similar agents playing the same role as the opponent. If agents belong to different groups, the group features could be unknown, but the proponent could use its experience with other agents of the opponent's group and infer them.

Moreover, either the proponent or the opponent's features could represent information about agents that act as representatives of a group and any agent can belong to different groups at the same time. In any case, the framework is flexible enough to work with this lack of knowledge, although the reliability of the conclusions drawn from previous experiences would be worse.

Also for simplicity, the chapter does not show how agents can use the dialogue graphs associated to argument-cases to take strategic decisions about which arguments are more suitable in a specific situation or about whether continuing with a current argumentation dialogue is worth. For instance, to improve efficiency in a negotiation an argumentation dialogue could be finished if it is being similar to a previous one that didn't reach an agreement. Else, opponent moves in a dialogue could be inferred by looking a similar previous dialogue with the same opponent. These issues will be dealt with in Chapter 4.

We have not presented in this chapter the storage of new cases and the maintenance of case-bases. This depends on the application domain and will be explained in detail latter with the study cases. In our AF, cases are stored at the end of the argumentation process, but not all cases are necessarily stored. As in most CBR systems, argument and domain-cases would be only stored if there is no similar enough case in the case-bases and the new domain and argumentation knowledge acquired must be stored. However, slightly different arguments could be represented with the same past argument-case by only updating its attacks information or attaching a new dialogue graph to its justification. Nevertheless, if the problem description, the acceptability of the argument or the conclusion change, a new argument-case has to be created. Also, to improve efficiency in searches, case-bases require a constant update to eliminate outdated cases, generalise and merge cases in a unique case when they are always indistinctly used, etc.

The actual algorithms to implement the agents' reasoning process depend on the application domain and the design of the real system that implements the framework. Also, the communication protocol that defines the dialogue, commitment and termination rules and the locutions that agents use to interchange arguments is presented

in the next chapter. These locutions depend on the agents' communication language and determine the intention of the argument (e.g. pose an attack or ask for justifications), the argument's sender and receiver, the format of the argument's content (e.g. if complete knowledge resources or parts are sent), etc. Also, the process to put critical questions depends on the actual ontology of argumentation schemes that agents implement and the interaction protocol that agents follow. These issues will be dealt with in the implementation of the case of study that will be proposed in Chapter 6.

Algorithm 1 generatePositions

Require: ProblemDescription, n, generation method (D, S, M) *//The description of the problem to solve, the maximum number n of positions to generate (all possible if n=0) and the method to generate positions*

matchCases = ∅
solutions = ∅
positions = ∅
argSchemes = ∅
extendedDesc = ∅
similarity = 0
SD = ∅
i = 0; j = 0; k = 0
if D ∈ GenerationMethod **then**
  for all $c \in DomainCasesCB$ **do**
    similarity = computeSimilarity(ProblemDescription, c)
    if $similarity > \delta$ **then**
      matchCases[i] = c *//If the similarity exceeds certain threshold, the domain-case is selected to generate the position*
      SD[i] = similarity *//The similarity degree of this domain-case is stored*
      i++
  solutions = generateSolutions(matchCases)
  if $lenght(solutions) \geq 1$ **then**
    for $[s = 0; s < lenght(solutions); s++]$ **do**
      positions = addPosition(ProblemDescription, solutions[s], SD[i])
if M ∈ GenerationMethod **then**
  for all $c \in DomainCasesCB$ **do**
    similarity = computeSimilarity(ProblemDescription, c)
    if $similarity > \delta$ **then**
      matchCases[i] = c *//If the similarity exceeds certain threshold, the domain-case is selected to generate the position*
      SD[i] = similarity *//The similarity degree of this domain-case is stored*
      i++
  extendedDesc = aggregateDescriptions(matchCases)
  if ProblemDescription ≠ extendedDesc **then**
    ProblemDescription = extendedDesc
    if S ∉ GenerationMethod **then**
      addGenerationMethod(S)
  else
    solutions = generateSolutions(matchCases)
  if $lenght(solutions) \geq 1$ **then**
    for $[s = 0; s < lenght(solutions); s++]$ **do**
      positions = addPosition(ProblemDescription, solutions[s], SD[i])
if S ∈ GenerationMethod **then**
  for all $as \in ArgumentationSchemesOnt$ **do**
    similarity = computeSimilarity(ProblemDescription, as)
    if $similarity > \eta$ **then**
      argSchemes[j] = k *//If the similarity exceeds certain threshold, the argumentation-scheme is selected to generate the position*
      SD[j] = similarity *//The similarity degree of this argumentation-scheme is stored*
      j++
  solutions = generateSolutions(argSchemes)
  if $lenght(solutions) \geq 1$ **then**
    for $[s = 0; s < lenght(solutions); s++]$ **do**
      positions = addPosition(ProblemDescription, solutions[s], SD[i])
Return positions

Algorithm 2 Position Generation and Selection

Require: ProblemDescription, generation method (D, S, M), w_{SimD}, w_{PD}, w_{SD}, w_{RD}, w_{AD}, w_{ED}, w_{EP} *// The description of the problem to solve, the generation method for generating cases from domain-cases (D), argumentation-schemes (S) or from both (M) and the weights for each element of the similarity degree and the support factor*

1: positions = $\emptyset$
2: argumentCases = $\emptyset$
3: SimD = $\emptyset$
4: SF = $\emptyset$
5: selectedPositions = $\emptyset$
6: positions = generatePositions(ProblemDescription, n)
7: **for** $[i = 1; i \leq lenght(positions); i++]$ **do**
8: argumentCases[i] = generateArgumentCase(ProblemDescription, positions[i])
9: SimD[i] = retrieveSimilarityDegree(positions[i])
10: **for** $[i = 1; i \leq lenght(argumentCases); i++]$ **do**
11: SF[i] = computeSF(ProblemDescription, argumentCases[i], argumenCases, w_{PD}, w_{SD}, w_{RD}, w_{AD}, w_{ED}, w_{EP})
12: selectedPositions = selectPosition(positions, argumentCases, SD, SF)
13: **Return** mostSuitable(selectedPositions)

Algorithm 3 computeSF

Require: ProblemDescription, argCase, argumentCases, w_{PD}, w_{SD}, w_{RD}, w_{AD}, w_{ED}, w_{EP} *//The description of the problem to solve, an argument case, the set of all argument-cases that represent all potential positions and the weights for each element of the support factor*

SF=0; PD=0; SD=0; RD=0; AD=0; ED=0; EP=0
att=0; minAtt=0; maxAtt=0; attAC=0
n=0; minS=0; maxS=0; stepsAC=0
kr=0; minKr=0; maxKr=0; krAC=0
arg = $\emptyset$
argC = $\emptyset$
argAccC = $\emptyset$
argAccCAtt = $\emptyset$
arg = retrieveSameProblem(argCase, ArgumentCasesCB)
argC = retrieveSameConclusion(argCase, ArgumentCasesCB)
argAccC = retrieveAccepted(argCase, ArgumentCasesCB)
argAccCAtt = retrieveAcceptedAttacked(argCase, ArgumentCasesCB)
if lenght(argC) $\neq$ 0 **then**
    PD = argAccC/argC
if lenght(arg) $\neq$ 0 **then**
    SD = argAccC/arg
if lenght(argC) $\neq$ 0 **then**
    RD = argAccCAtt/argC
att = computeNumberOfAttacks(argCase)
minAtt = att
maxAtt = att
for all [$ac \in argumentCases$] **do**
    attAC = computeNumberOfAttacks(ac)
    if minAtt > attAC **then**
        minAtt = attAC
    if maxAtt < attAC **then**
        maxAtt = attAC
if maxAtt $\neq$ minAtt **then**
    AD = (att - minAtt)/(maxAtt - minAtt)
n = computeNumberOfSteps(argCase)
minS = n
maxS = n
for all [$ac \in argumentCases$] **do**
    stepsAC = computeNumberOfSteps(ac)
    if minS > stepsAC **then**
        minS = stepsAC
    if maxS < stepsAC **then**
        maxS = stepsAC
if maxS $\neq$ minS **then**
    ED = 1 - ((n - minS)/(maxS - minS))
kr = computeNumberOfKR(argCase)
minKr = kr
maxKr = kr
for all [$ac \in argumentCases$] **do**
    krAC = computeNumberOfKR(ac)
    if minKr > krAC **then**
        minKr = krAC
    if maxKr < krAC **then**
        maxKr = krAC
if maxKr $\neq$ minKr **then**
    EP = (kr - minKr)/(maxKr - minKr)
Return $SF = (w_{PD} * PD + w_{SD} * SD + w_{RD} * (1 - RD) + w_{AD} * (1 - AD) + w_{ED} * ED + w_{EP} * EP)$

Algorithm 4 Position Evaluation

Require: position, incPosition, positions *//The position of an agent, an incoming position and the set of potential positions generated by the agent*

```
1: dependencyR = checkDependencyRelation(opponent, proponent)
2: if (dependencyR =! "Power") && (dependencyR =! "Authorisation") then
3:    if position = incPosition then
4:       acceptPosition(incPosition) //If the positions are the same, the agent accepts
         the incoming position
5:    if (position ≠ incPosition) && (incPosition ∈ positions) then
6:       decideAttack(incPosition)
7:    if incPosition notin positions then
8:       askForSupport(incPosition)
9: else
10:   acceptPosition(incPosition) //If the opponet agent has a power or an authorisa-
      tion relation over the proponent, it changes its position and accepts the incoming
      position
```

Algorithm 5 Argument Generation

Require: position, incArgument *//The position of the agent and the incoming argument*

```
1: requestType = evaluateIncomingRequest()
2: if requestType = supportAsked then
3:    decideSupport(position)
4: if requestType = attackReceived then
5:    evaluateArgument(incArgument)
```

Algorithm 6 generateAttack

Require: incArgument, DomainCasesCB, ArgumentCasesCB, ArgumentationSchemes //*The argument to attack, the case-bases of domain-cases and argument-cases and the argumentation-schemes ontology*

```
1: support = checkSupportSet(incArgument)
2: supportElement = selectElementToAttack(support)
3: if supportElement = Premises then
4:    generateDPAttack(incArgument)
5: if supportElement = ArgumentationSchemes then
6:    if ArgumentationScheme ∈ ArgumentationSchemesOnt then
7:       generateASAttack(incArgument)
8:    else
9:       storeAS(ArgumentationScheme)
10: if (supportElement = Domain-Case) or (supportElement = Argument-Case) then
11:    CE=generateCounterExample(incArgument)
12:    if CE ≠ ∅ then
13:       generateCEAttack(incArgument)
14:    else
15:       generateDPAttack(incArgument)
16: if support = ∅ then
17:    generateCEAttack(incArgument)
```

Algorithm 7 evaluateArgument

Require: incArgument //*The incoming argument to evaluate*

```
1: if defeats(currArgument, incArgument) then
2:    generateAttack(incArgument)
3: else
4:    newS = generateNewSupport(position)
5:    if newS == 0 then
6:       withdraw(position)
```

Chapter

4

Dialogue Game Protocol

In this chapter we present the communication protocol that agents of our argumentation framework use to interact when they engage in argumentation dialogues. Considerable research has been performed on the design of artificial agent communication languages, such as DARPA's *Knowledge Query and Manipulation Language (KQML)*[1] and the IEEE Foundation for Intelligent Physical Agents *Agent Communications Language (FIPA ACL)*[2]. These languages provide agents with a 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 unending dialogues [McBurney and Parsons, 2009, chapter 13]. A possible solution for this problem consists in limiting the allowed set of utterances for each step of the dialogue by defining the agents communication protocol with a dialogue game. As shown in Chapter 2, formal dialogue games are interactions between several players (agents in our case) where each player moves by making utterances in accordance to a defined set of rules [McBurney and Parsons, 2002a].

[1] `www.cs.umbc.edu/research/kqml/`
[2] `www.fipa.org/repository/aclspecs.html`

The structure of this chapter is as follows: first, Section 4.1 introduces the notation used along the chapter; Section 4.2 presents the syntax of the protocol, as the set of defined locutions, their combinatorial properties and the rules that govern the dialogue; After that, Section 4.3 defines several dialogue strategies that can follow the agents of our case-based argumentation framework to take advantage over other participants of the dialogue; Then, Section 4.4 provides the *axiomatic* semantics and the *operational* semantics of the locutions. The former defines the pre-conditions that should be met to put forward each locution (or set of locutions) and the post-conditions that apply before their utterance. The latter views each locution as a transition in an abstract state-machine that represents the possible stages that can be reached during the dialogue; Finally, section 4.5 summarises the contents of this chapter.

4.1 Preliminaries

Along this chapter 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 used in the ArgCBROnto ontology to define the semantics of concepts and relations between concepts. On the other hand, we use the standard operators and axioms of *modal logics* of knowledge and believe [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 particular DL. The semantics of DLs is given in terms of *interpretations* [Baader et al., 2007]:

Definition 4.1.1 (Interpretation:) *An interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$*, called the domain of* $\mathcal{I}$*, and a function* $\cdot^{\mathcal{I}}$ *that maps every concept to a subset of* $\Delta^{\mathcal{I}}$*, and every role name to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *such that, for all concepts* C*,* D *and all role names* R*,*
$\top^{\mathcal{I}} = \{\Delta^{\mathcal{I}} |$ *for all* $C^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*, then* $\top^{\mathcal{I}} = C^{\mathcal{I}} \cup \neg C^{\mathcal{I}}\}$,

$\bot^{\mathcal{I}} = \{\emptyset |$ *for all* $C \in \Delta^{\mathcal{I}}$*, then* $\bot^{\mathcal{I}} = C^{\mathcal{I}} \cap \neg C^{\mathcal{I}}\}$,
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |$ *there is some* $y \in \Delta^{\mathcal{I}}$ *with* $\langle x, y\rangle \in R^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}\}$,
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |$ *for all* $y \in \Delta^{\mathcal{I}}$*, if* $\langle x, y\rangle \in R^{\mathcal{I}}$*, then* $y \in C^{\mathcal{I}}\}$. *We say that* $C^{\mathcal{I}}$ *(*$R^{\mathcal{I}}$*) is the extension of the concept* C *(role name* R*) in the interpretation* $\mathcal{I}$*. If* $x \in C^{\mathcal{I}}$*, then we say that* x *is an instance of* C *in* $\mathcal{I}$.

Table 4.1 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*.

As any 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: a *terminological (TBox)* and an *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, as pointed out in Chapter 3, 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:

$$SocialEntity \sqsubseteq Thing$$

$$Agent \sqsubseteq SocialEntity$$

$$Argument \sqsubseteq Thing$$

Furthermore, 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 say that arguments can have three properties that relate them with 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 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 ϕ is the content of the utterance. The former locution is addressed to all participants in the dialogue, whereas the latter 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, evidences 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, Φ is said to be an argument in support of ϕ if $\Phi \in \mathcal{A}/\Phi \vdash^{+} \phi$. Correspondingly, Φ is said to be an argument against ϕ 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 some locution and other agent challenges it, the first has the commitment to provide reasons (or arguments) to justify the validity of such assertion or else, 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 before, we follow the standard notation of *modal logics* of knowledge and believe described in [Shoham and Leyton-Brown, 2009, chapter 13]. Thus, we use the modal operators

$K_i\phi$: "Agent a_i knows ϕ"

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

$C_g\phi$: "ϕ 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 ϕ 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 can be true or not).For instance, all agents that belong to a society that represents the workers of a car rental company can *know* that the business manager *believe* that Volvo is the safest brand that they can offer to its customers. The workers know what the manager believe, and the fact that everybody knows the opinion of the manager is true. However, this doesn't mean that such opinion has to be true and in fact, any worker can believe that BMW is safer than Volvo. Therefore, the belief of the manager is subjective and depends on its individual knowledge.

In addition, as proposed in [McBurney and Parsons, 2004], we use the following simplified elements of *FIPA*'s communicative act library specification[5]:

Done[locution(a_s, ϕ)*, preconditions]*

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

Feasible[condition, locution(a_s, ϕ)*]*

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

Further notation that we use throughout this chapter is the next:

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

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

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

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 Syntax

In this section we provide the syntax of the communication protocol that follow the agents of our argumentation framework. To formalise it, 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 modeling systems to represent the reality and that do not exist yet), which fits the objective of most open MAS. Other approaches to formalise dialogue systems are reviewed in [Prakken, 2006] (concretely, 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 potentially partial knowledge about the context of the dispute can enter or leave the system (and hence the dialogue) at any time.

Next, 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 seen as a collaborative deliberation, where all agents follow to 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 apply, or also as a negotiation, where agents try to convince other agents to apply its solution as the best for solving the problem and have individual *utility functions* that increase its perceived utility in that case.

Locutions

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

L1: *open_dialogue*(a_s, ϕ), where ϕ 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 it has been presented with.

L2: *enter_dialogue*(a_s, ϕ), where ϕ 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, ϕ), where ϕ 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.

L4: *propose*(a_s, ϕ), where ϕ 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, ϕ), where ϕ 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 support argument.

L6: *noCommit*(a_s, ϕ), where ϕ 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, ϕ), where ϕ can be an argument $arg \in \mathcal{A}$ that supports a position, other 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, ϕ), where ϕ 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 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 ϕ 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 ϕ 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 found 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, it 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 4.1 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. Next, the stages of our dialogue game and the rules for the combination of locutions in each stage are presented.

Opening Stage:

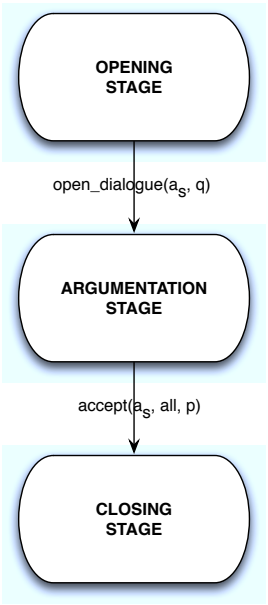

Figure 4.1: State Machine of the Dialogue Game

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:

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 4.2, this stage is divided into a set of substages which activation is defined by the following rules (for clarity reasons, 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 being proposed its own position) with the locution $why(a_s, a_r, p)$. Also, in this substage the agent can withdraw from the dialogue by using the locution

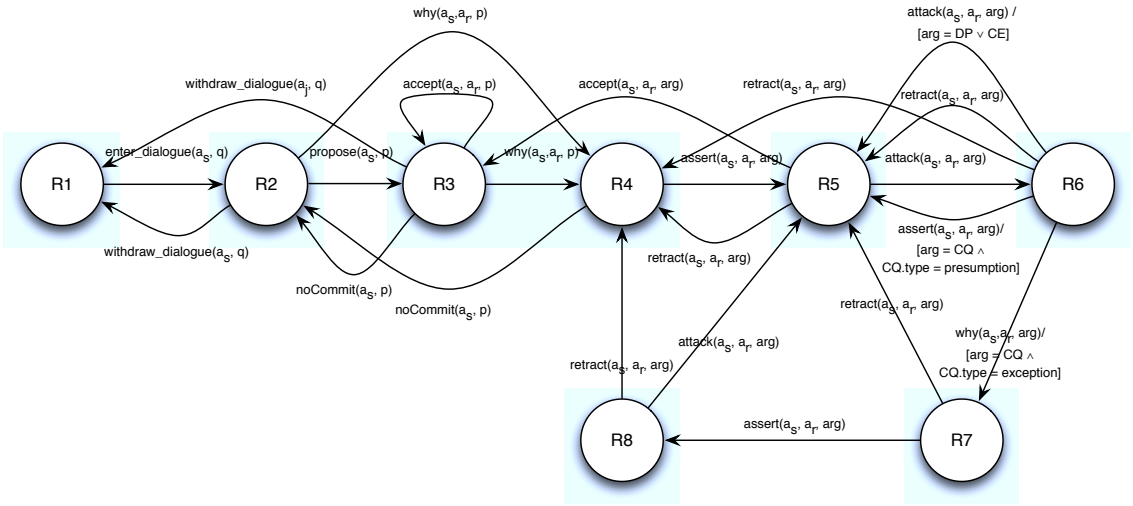

Figure 4.2: State Machine of the Argumentation Stage

$withdraw\_dialogue(a_s, q)$.

R3: In this substage, an agent that has proposed its position p to solve the problem q can be asked by other 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, which reports 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 other 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 support argument and report to the other agent with the locution $accept(a_s, a_r, arg)$, where arg is the support 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 other 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 other 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 to the other agent an argument arg with an 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 such 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 other agent for providing a support 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 an evidence to support such 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 other agent with an evidence that supports the other agent's critical question of the type exception, this agent can retract its argument arg and report to the other 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 are a transaction labelled with the locution $noCommit(a_s, p)$ from substages $R5...R8$ to substage $R2$, although they do not appear in Figure 4.2 for clarity purposes.

Closing Stage:

The closing stage can be activated at any time of 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

$a_s = a_i$) that informs all participating agents about the final position p agreed as the solution for the problem q. Here, the commitment store of all agents is deleted.

Commitment Rules

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

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.

Then, 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.

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. Then, 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 before, commitment stores cannot have inconsistent arguments. Then, 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 is a critical question of the type presumption, the locution $attack(a_s, a_r, arg)$ commits the receiver to providing an argument as evidence to support its last argument or else to retracting it.

CR11 : The locution $why(a_s, a_r, arg)$ where arg is an attack argument of the type *exception* put forward by the receiver to the last argument of the sender commits the receiver to having an argument to support its challenge or else to retracting it.

CR12: The locution $accept(a_s, all, p)$ $(all : \forall a_i, a_i \neq a_s / Agent(a_i))$ deletes the commitment stores of all agents that are still participating in the dialogue (including the initiator).

Rules for Speaker Order

During the dialogue, agents take turns to put forward locutions. Each time an agent a_s sends a locution to other agent a_r, it waits for an answer from a_r. However, any agent can held parallel argumentation dialogues with several agents. Then, in each of these dialogues, the argumentation succeeds as a two party dialogue between two agents, the one sending a locution to the other 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 agents that are engaged in the dialogue. Also, with these locutions, the sender agent does not wait for any response.

In this dialogue game protocol, we assume that all participating agents are able to see at each time 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 access to the commitment store of the other. In this way, these agents can see the commitments associated to the arguments of their partners, but other agents can only have access to the positions proposed by each agent in the dialogue (also stored in the commitment stores). This preserves the privacy of the arguments that an agent puts forward in its argumentation dialogue with other agent. Note that if an agent wants to ask other agents for an opinion about an argument that it has received, it only has to send to these agents the argument, as it was its own argument. This simple rule allow us to use the same dialogue game to govern both collaborative deliberations and negotiations. In the former, all agents follow the common objective of proposing the best solution for a problem at hand. Therefore, there are not interested agents trying to take profit of the information interchanged between other agents to get a greater benefit with the final agreement reached. However, it could be the case in a negotiation, where each agent tries to increase its utility value perceived with the final agreement and use any extra information about other agents' knowledge and preferences 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 other 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 are 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 all participating agents the final position p 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 other agent. 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 the substage $R2$ or listening to incoming locutions of other agents in the substage $R3$. Then, the specific dialogue with an agent that has asked other agent for a support argument for its position p follows to the subsequent substages, but the agent still remains listening in $R3$ to other requests. Finally, the locution $noCommit(a_s, p)$ commits the sender to terminate any dialogue that its has started to defend p.

4.3 Dialogue Strategies

As pointed out in [Amgoud and Hameurlain, 2006], there is no consensus on the definition of a strategy and on the parameters necessary for its definition. Consequently, there are no standard methodology and no formal models for strategies. A first attempt to model the process of strategy construction for negotiation dialogues was published in [Rahwan et al., 2007a]. This work proposes a methodology for designing heuristic negotiation strategies that guides the strategy designer along several stages to produce modular specifications of tactics and strategies.

The literature on strategies for argumentation provides different definitions for the notion of strategy. Several examples are:

> "The strategy is a decision problem in which an agent tries to choose among different alternatives the best option, which according to its beliefs, will

satisfy at least its most important goals [Amgoud and Hameurlain, 2006]."

"Private strategies, as adopted by an individual agent, specify the dialogue move(s) the agent is willing to utter, according to its own objectives and other personal characteristics.[Kakas et al., 2005]"

"A strategy of an agent specifies a complete plan that describes what action the agent takes for every decision that a player might be called upon to take, for every piece of information that the player might have at each time that it is called upon to act. Thus a strategy for an agent would specify for each possible subset of arguments that could define its type (the set of arguments that the agent is capable of putting forward), what set of arguments to reveal.[Rahwan and Larson, 2009, Chapter 16]"

In this section we propose several dialogue strategies for different types of agent profiles (agent attitudes) in our case-based argumentation framework. In doing so, we are closer to the heuristic approach of [Amgoud and Hameurlain, 2006] rather than to game theoretic one. There are several reasons behind this decision. One one hand, game theoretic approaches are usually applied to abstract argumentation frameworks where the strategies of agents determine which argument(s) they will reveal in each argumentation step. Common objectives of these works are to study the conditions under which the outcome of the game is not affected by the strategic decisions of the agents or to predict the outcome of the game. In contrast, we define dialogue strategies on basis of the knowledge that agents have about the domain, previous argumentation experiences and the social context (the roles, norms, preferences over values and dependency relations among agents and groups). In doing so, we take into account the specific structure of arguments and the knowledge resources of our framework and design strategies to help agents to take advantage over other participants in the argumentation dialogue.

On the other hand, in our application domain there is not a pre-defined utility function about the payoff that an agent gets for the fact of winning the dialogue or having accepted more or less arguments, which is one of the common assumptions in game theoretic approaches for strategic argumentation. Finally, game theory assumes complete knowledge of the space of arguments proposed in the argumentation framework. This assumption is unrealistic in an argumentation dialogue between heterogeneous agents which have individual and private knowledge resources to generate arguments.

Further disadvantages of applying game theory to devise dialogue strategies in argumentation frameworks are the same as those reported in [Jennings et al., 2001] for the problem of applying game theory to automated negotiation. First, game theoretic studies of rational choice in multi-agent encounters typically assume that agents are allowed to select the best strategy from the space of all possible strategies, by considering all possible interactions. This is computationally intractable when the space of possible choices growths. Also, game theory assumes that it is possible to characterise an agent's preferences about possible outcomes. This is hard to define for agents that represent humans that are engaged in an argumentation process. Our alternative solution is to define preferences over values instead of preferences over dialogue outcomes and use them to guide the agents choices.

In each step of the dialogue of our case-based argumentation framework an agent would choose a specific locution and a content for it depending on its profile and the strategy that follows. Assuming that L represents the set of available locutions of the dialogue game, let us suppose that the function *Replies* returns for each locution the legal replies to it:

$$Replies : L \rightarrow 2^L$$

Recalling that D represents the set of well-formed formulae in our content language $SHOIN(D)$, the function *Content* returns for a given locution, the set of possible contents:

$$Content : L \rightarrow 2^D$$

In each step of the argumentation dialogue, agents exchange moves.

Definition 4.3.1 (Move) *A move is a pair (l, ϕ), where $l \in L$ and $\phi \in$ Content(l).*

Thus, the strategy problem is formalised as in [Amgoud and Hameurlain, 2006]:

Definition 4.3.2 (Strategy Problem) *Let (l, ϕ) be the current move in a dialogue. What is the next move (l', ϕ') to utter such that l' $\in$ Replies(l)?*

The answer for this question implies to find the best locution and content that the agent can utter in each step of the dialogue, given the profile of the agent and its knowledge. Therefore, a dialogue strategy is defined as follows:

Definition 4.3.3 (Dialogue Strategy) *A dialogue strategy is defined as a function* $S: 2^{L \times D} \rightarrow L \times D$ *where* $(l, \phi) \in L \times D$.

Given a move $m=(l, \phi)$, $S(m) = m'$ such that $m' = (l', \phi')$ is the best move that an agent can utter in the next step of the dialogue taking into account its profile and experience. In our case-based argumentation framework, agents select the best locution to bring up depending on their profile and the content of this locution depending on the knowledge that they have in their knowledge resources and the tactic that they follow.

We consider the following agent profiles [Amgoud and Parsons, 2001]:

- Agreeable: accept whenever possible.
- Disagreeable: only accept when there is no reason not to.
- Open-minded: only attack when necessary.
- Argumentative: attack whenever possible.
- Elephant's child: challenge whenever possible.

An agreeable agent will initially accept positions and arguments from peers (other agents that it has a charity dependency relation with them) whenever is possible. This means that it will accept any position that is in the list of its potential positions (even if it is not ranked as the most suitable) and it will accept any argument from a peer if it does not have a counter-argument, a distinguishing premise or a critical question that attack it. Therefore, if the agent cannot rebut an argument from a peer, it will accept it and also its associated position. Agreeable agents do not challenge positions of other agents, but just try to defend theirs if attacked. In the case that an agreeable agent cannot generate a position, it does not participate in the dialogue.

A disagreeable agent will initially accept the position of a peer if it is ranked first in its list of potential positions. Regarding arguments, this type of agent will try to generate an attack to any argument that it receives from other agents. If it is not able to generate such an attack, the agent will accept the argument of its peer, but it still will not accept the peer's position. Disagreeable agents do not challenge positions of other agents, but just try to defend theirs if attacked. In the case that a disagreeable agent cannot generate a position, it does not participate in the dialogue.

An open-minded agent challenges different positions of other peers. Also, it waits for challenges from other peers and will try to rebut their attack arguments. If a peer wins the discussion, this type of agent accepts its argument and its associated position. If it cannot generate positions, it does not engage in the dialogue.

An argumentative agent will not initially accept any position from a peer. This type of agent will challenge positions of other peers when they are different from its position, even if they appear in its list of potential positions to propose. Also, it will try to generate an answer for any attack that it receives, but opposite to open-minded agents, argumentative agents do not accept the position of the peer that generated the attack if the last wins the debate. If an argumentative agent cannot generate positions, it will not participate in the dialogue.

An elephant's child agent will always challenge the positions of other peers (even if they have proposed the same position than it or if it cannot generate positions). If it can generate attacks, it will put forward them to rebut the arguments of other agents, but if they win the debate, this type of agent does not accept their positions. In fact, the only way an elephant's child agent accepts the position of other agent is when it attacks this position and the attacked agent wins the debate.

Independently of their profile, agents will accept arguments from other agents that have a power or authorisation dependency relation over them. Recall that in any case the acceptance of an argument is subjected to the defeat relation defined in the argumentation framework. Table 4.2 summarises the behaviour of these agent profiles.

The legends of the columns of Table 4.2 are the following:

- **ASP (Accept Same Position):** Initially accept the position of a peer that matches the agent's current position.
- **APL (Accept Position in List):** Initially accept the position of a peer that is in the agent's list of potential positions (although not ranked as the most suitable position to propose).
- **CSP (Challenge Same Position):** Challenge the position of a peer that has proposed the same position than the agent's one.
- **CPL (Challenge Position in List):** Challenge the position of a peer that has proposed a position that is in the agent's list of potential positions (although not ranked as the most suitable position to propose).

- **CDP (Challenge Different Position):** Challenge the position of peer that has proposed a different position (and this position is not in the agent's list of potential positions to propose).
- **AAP (Accept Attacked Position):** When the agent has held a discussion with other agent and the later wins the debate, accept its position.
- **AP$\emptyset$ (Accept other Positions if no position can be generated).**
- **CP$\emptyset$ (Challenge other Positions if no position can be generated).**

Depending on its profile, the agent will choose the next locution to put forward on the dialogue game. Then, for instance, in an agent a_i has proposed a position q to solve the problem p under discussion (*propose*(a_s, q)), an agreeable agent a_j that entered in the dialogue (*enter_dialogue*(a_j, p)) and could not generate its own position would accept q. Thus, S_{a_j}(*enter_dialogue, Problem(p)) = (accept, Solution(q))*. However, let us assume that a_j has proposed a position q' and that a_i has challenged it (*why*(a_i, a_j, *q')*). In this case, the agreeable agent a_j will try to find a support argument for its position by searching its domain and argument-cases case-bases. Now, S_{a_j}*(enter_dialogue, Problem(p)) = (assert, ϕ)*. Then, among the potential arguments that a_j could find, it has to select one to support the position. This implies to select the content ϕ of the locution *assert*. To make this selection, agents can use the following tactics, which consist on assigning more or less weight to the elements of the support factor used to select positions and arguments:

- Persuasive Tactic: the agent selects such positions and arguments whose associated argument-cases were more persuasive in the past (have more persuasiveness degree).
- Maximise-Support Tactic: the agent selects such positions and arguments that have higher probability of being accepted at the end of the dialogue (their associated argument-cases have more support degree).
- Minimise-Risk Strategy: the agent selects such positions and arguments that have a lower probability of being attacked (their associated argument-cases have less risk degree).
- Minimise-Attack Tactic: the agent selects such positions and arguments that have received a lower number of attacks in the past (their associated argument-cases have less attack degree).

- Maximise-Efficiency Tactic: the agent selects such positions and arguments that lead to shorter argumentation dialogues (their associated argument-cases have higher efficiency degree).
- Explanatory Tactic: the agent selects such positions and arguments that cover a bigger number of domain-cases or argumentation schemes. That is, the positions and arguments that are similar to argument-cases that have more justification elements (more domain-cases or argumentation schemes in the justification part).

Both profile and tactic form the strategy of an agent. The performance of different strategies will be evaluated in the study case proposed in Chapter 6.

4.4 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 to provide a communication language with a semantics [Tennent, 1991], for instance, the *axiomatic* approach and the *operational* approach. The semantics of the locutions that define the communication language of the dialogue game presented in this chapter are provided in the following sections.

4.4.1 Axiomatic Semantics

The basic approach of semantics for communication languages is the axiomatic approach. With this approach, the meaning of the language is not explicitly defined but given in terms of properties that the language concepts satisfy [van Eijk, 2002]. Thus, in axiomatic semantics the semantics of a locution L is defined as a triple:

$$\{pre\}\ L\ \{post\}$$

where *pre* represents the preconditions that must hold before the locution is uttered and *post* represents the postconditions that apply after this utterance. Also, we can distinguish between *private* axiomatic semantics, where some preconditions or postconditions describe conditions of the dialogue that can only be observed by some agents and *public* axiomatic semantics, where all conditions are accessible to all agents. In our

dialogue game protocol, the preconditions and postconditions of some locutions can only be observed by the sender and receiver agents. Thus, we present in this section the private axiomatic semantics for the locutions of our dialogue game. For clarity purposes, the preconditions that hold before the utterance of each locution in the communicative act *Done* are assumed to be the preconditions specified in the axiomatic semantics definition of the locution and thus, are omitted in the text.

Locutions Axiomatic Semantics

- **{*pre*}** $open\_dialogue(a_s, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ wants to inform other agents of the society S_t (as it was defined in Chapter 3) about the proposition ϕ, which is a $Problem(q)$ to solve. Note that until agents do not enter the dialogue, their commitment stores CS are not created.
  $(K_s q) \wedge (\nexists CS_s)$

  post : All agents in the society S_t know that $Agent(a_s)$ wants to solve $Problem(q)$.
  $(\Diamond C_{S_t} K_s q)$

- **{*pre*}** $enter\_dialogue(a_s, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ knows ϕ (the $Problem(q)$ reported by $Agent(a_i)$) and informs other participants of the $Group(g)$[6] that are engaged in the dialogue that it is willing to enter the dialogue to solve $Problem(q)$.
  $(Done[open\_dialogue(a_i, q), ...]) \wedge (K_s q)$

  post : Other participants of the $Group(g)$ are informed that $Agent(a_s)$ is willing to engage in a dialogue to solve $Problem(q)$. Also, the commitment store CS_s is created and $Agent(a_s)$ starts to belong to the $Group(g)$ of agents engaged in the dialogue.
  $(\Diamond C_g(a_s \in g)) \wedge (\exists CS_s)$

- **{*pre*}** $withdraw\_dialogue(a_s, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ that has engaged in the argumentation dialogue to solve ϕ (the $Problem(q)$) wants to leave from the dialogue and report it to the other agents of the $Group(g)$ that are engaged in the dialogue. Note that agents

[6] Agents know which other agents are participating in the dialogue by looking at their commitment stores.

cannot withdraw the dialogue before withdrawing any position $Solution(p)$ that they have proposed.
$(Done[enter\_dialogue(a_s, q), K_s q]) \wedge (\nexists p \in CS_s)$

post : Other participating agents of the $Group(g)$ know that $Agent(a_s)$ no longer participates in the dialogue to solve $Problem(q)$. Also, the commitment store CS_s of $Agent(a_s)$ is deleted.
$(\Diamond C_g(a_s \notin g)) \wedge (\nexists CS_s)$

- **{*pre*}** $propose(a_s, \phi)$ **{*post*}**

  pre : An $Agent(a_s)$ that has engaged in a dialogue to solve ϕ (the $Problem(q)$) wants to propose its position $Solution(p)$ as a solution for the problem and reports it to the other agents of the $Group(g)$. An agent cannot propose a new position without withdrawing a previous $Solution(r)$ from its commitment store, if any.
  $(Done[enter\_dialogue(a_s, q), ...]) \wedge (B_s p) \wedge (\forall r \neq p)(\nexists r \in CS_s)$

  post : Other participating agents of the $Group(g)$ know that $Agent(a_s)$ has proposed position $Solution(p)$ as solution for $Problem(q)$ and it is inserted in the commitment store CS_s of $Agent(a_s)$.
  $(\Diamond C_g B_s p) \wedge (p \in CS_s)$

- **{*pre*}** $why(a_s, a_r, \phi)$ **{*post*}**
  This locution has different semantics depending on its content ϕ. On one hand, if ϕ is a position $Solution(p)$ to solve the problem under discussion we have the following conditions:

  pre : $Agent(a_s)$ wants to challenge $Agent(a_r)$ to provide a justification for the position $Solution(p)$.
  $(Done[propose(a_r, p), B_r p]) \wedge (K_s B_r p) \wedge (p \notin CS_s)$

  post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe $Solution(p)$ and has the dialogical commitment of justifying it with an $Argument(arg) \vdash^+ Solution(p)$ or else of withdrawing it.
  $(\Diamond K_r \neg B_s p) \wedge ((Feasible[\exists arg / arg \vdash^+ p), assert(a_r, a_s, arg)]) \vee$
  $(Feasible[\nexists arg / arg \vdash^+ p), noCommit(a_r, p)]))$

  On the other hand, if ϕ is an attacking $Argument(arg_r)$ of $Agent(a_r)$ that poses a critical question $Premise(exc)$ of the type *exception* to a previous $Argument(arg_s)$

of $Agent(a_s)$ (such that $hasSupportSet(arg_r, SS_r) \wedge hasException(SS_r, exc) \wedge Argument(arg_r) \vdash^{-} Argument(arg_s)$) we have the following conditions:

pre : $Agent(a_s)$ wants to challenge $Agent(a_r)$ to provide a justification *Argument($arg_{r'}$)* for its attack $Argument(arg_r)$ to $Argument(arg_s)$.
$(Done[attack(a_r, a_s, arg_r), \neg B_r arg_s]) \wedge (K_s B_r arg_r) \wedge (arg_r \notin CS_s)$

post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe its $Argument(arg_r)$. Thus, it is committed to providing a justification $Argument(arg_{r'})$ for its attacking argument $Argument(arg_r)$ such that $Argument(arg_{r'}) \vdash^{+}$ *Argument(arg_r)* or else, to withdrawing the it.
$(\Diamond K_r \neg B_s arg_r) \wedge ((Feasible[\exists arg_{r'}/arg_{r'} \vdash^{+} arg_r), assert(a_r, a_s, arg_{r'})]) \vee (Feasible[\nexists arg_{r'}/arg_{r'} \vdash^{+} arg_r), retract(a_r, a_s, arg_r)]))$

- **{*pre*}** $noCommit(a_s, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ that has put forward ϕ (the position $Solution(p)$) wants to withdraw it and report this change to the other participating agents of the $Group(g)$ engaged in the dialogue.
  $(p \in CS_s) \wedge (C_g B_s p)$

  post : Other participating agents of the $Group(g)$ know that $Agent(a_s)$ no longer proposes $Solution(p)$ as its position to solve the problem at hand. Also, $Solution(p)$ is deleted from the commitment store CS_s of $Agent(a_s)$.
  $(\Diamond C_g \neg B_s p) \wedge (p \notin CS_s)$

- **{*pre*}** $assert(a_s, a_r, \phi)$ **{*post*}**
  This locution has different semantics depending on its content ϕ. However, in any case an argument cannot be inserted in the commitment store of an agent without deleting first any inconsistent argument. Then, on one hand, it could be the case that $Agent(a_r)$ has challenged the position $Solution(p)$ of $Agent(a_s)$. In this case, ϕ is an $Argument(arg)$ that supports this position, such that $Argument(arg) \vdash^{+} Solution(p)$.

  pre : An $Agent(a_s)$ wants to provide a justification for its position $Solution(p)$ and reports it to an agent $Agent(a_r)$ that has challenged it.
  $(Done[why(a_r, a_s, p), ...]) \wedge (\exists arg)(arg \vdash^{+} p) \wedge (\nexists arg_s \in CS_s)(arg \vdash^{-} arg_s)$

  post : $Agent(a_r)$ knows that $Agent(a_s)$ has provided $Argument(arg)$ as a justification for its position and it is inserted in the commitment store CS_s of

$Agent(a_s)$.
$(\Diamond K_r B_s arg) \wedge (arg \in CS_s)$

On the other hand, $Agent(a_r)$ could have attacked the argument $Argument(arg_s)$ with an $Argument(arg_r)$ that poses a critical question $Premise(pre)$ of the type *presumption* such that $hasSupportSet(arg_r, SS_r) \wedge hasPresumption(arg_r, pre) \wedge Argument(arg_r) \vdash^- Argument(arg_s)$. In this case, ϕ is an $Argument(arg_{s'})$ of $Agent(a_s)$ that supports its previous argument $Argument(arg_s)$, such that $Argument(arg_{s'}) \vdash^+ Argument(arg_s)$.

pre : $Agent(a_s)$ wants to rebut a critical question attack of the type *presumption* posed by $Agent(a_r)$, such that $Argument(arg_r) \vdash^- Argument(arg_s)$, but $Argument(arg_{s'}) \vdash^+ Argument(arg_s)$ and hence rebuts $Argument(arg_r)$.
$Done[attack(a_r, a_s, arg_r), ...] \wedge (\exists arg_{s'})(arg_{s'}$
$\vdash^+ arg_s) \wedge (\nexists arg_{s''} \in CS_s)(arg_{s'} \vdash^- arg_{s''})$

post : $Agent(a_r)$ knows that $Agent(a_s)$ has provided $Argument(arg_{s'})$ as a justification for its $Argument(arg_s)$ and it is inserted in the commitment store CS_s of $Agent(a_s)$.
$(\Diamond K_r B_s arg_{s'}) \wedge (arg_{s'} \in CS_s)$

Finally, $Agent(a_r)$ could have challenged the argument $Argument(arg_s)$ that poses a critical question $Premise(exc)$ of the type *exception* to its $Argument(arg_r)$ such that $hasSupportSet(arg_s, SS_s) \wedge hasException(SS_s, exc) \wedge Argument(arg_s)$ $\vdash^- Argument(arg_r)$. In this case, ϕ is an $Argument(arg_{s'})$ of $Agent(a_s)$ that supports its critical question attack posed with $Argument(arg_s)$, such that $Argument(arg_{s'}) \vdash^+ Argument(arg_s)$.

pre : $Agent(a_s)$ wants to support a critical question attack of the type *exception*, such that $Argument(arg_s) \vdash^- Argument(arg_r)$ and $Argument(arg_{s'}) \vdash^+ Argument(arg_s)$.
$Done[why(a_r, a_s, arg_s), ...] \wedge (\exists arg_{s'})(arg_{s'} \vdash^+ arg_s) \wedge$
$(\nexists arg_{s''} \in CS_s)(arg_{s'} \vdash^- arg_{s''})$

post : $Agent(a_r)$ knows that $Agent(a_s)$ has provided $Argument(arg_{s'})$ as a justification for its $Argument(arg_s)$ and it is inserted in the commitment store CS_s of $Agent(a_s)$.
$(\Diamond K_r B_s arg_{s'}) \wedge (arg_{s'} \in CS_s)$

- **{*pre*}** $attack(a_s, a_r, \phi)$ **{*post*}**
  This locution has different semantics depending on its content ϕ, which represents different types of arguments. Again, in any case an argument cannot be inserted in the commitment store of an agent without deleting first any inconsistent argument. With the *attack* locution, the $Agent(a_s)$ puts forward an $Argument(arg_s)$ to attack the $Argument(arg_r)$ of an $Agent(a_r)$, such that $Argument(arg_s) \vdash^{-} Argument(arg_r)$. $Argument(arg_s)$ can be of different types. On one hand, if $Argument(arg_r)$ is a support argument with one or more premises in its support set, such that $hasSupportSet(arg_r, SS_r) \wedge Premise(pr_r) \wedge hasPremise(SS_r, pr_r)$, $Argument(arg_s)$ can be a *distinguishing-premise* attack.

  pre : $Agent(a_s)$ wants to attack the support $Argument(arg_r)$ of an $Agent(a_r)$ with an $Argument(arg_s)$, such that $hasSupportSet(arg, SS_s) \wedge Premise(DP) \wedge hasDistinguishingPremise(SS_s, DP)$. $(Done[assert(a_r, a_s, arg_r), ...]) \wedge (\exists arg_s)(arg_s \vdash^{-} arg_r) \wedge (\nexists arg_{s'} \in CS_s)$ $(arg_s \vdash^{-} arg_{s'})$

  post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe its support $Argument(arg_r)$ and $Argument(arg_s)$ is inserted into the commitment store CS_s of $Agent(a_s)$. $(\Diamond K_r \neg B_s arg_r) \wedge (K_r B_s arg_s) \wedge (arg_s \in CS_s)$

  On the other hand, if $Argument(arg_r)$ is a support argument with one or more *argument-cases* or *domain-cases* in its support set, such that $hasSupportSet(arg_r,\ SS_r) \wedge Case(c_r) \wedge (hasDomainCase(SS_r, c_r) \vee hasArgumentCase(SS_r, c_r))$, then $Argument(arg_s)$ can be a *counter-example* attack. In that case, the axiomatic semantics coincide with the previous case.

  Alternatively, if $Argument(arg_r)$ is a support argument with one or more *argumentation-schemes* in its support set, such that $hasSupportSet(arg_r, SS_r) \wedge ArgumentationScheme(as_r) \wedge (hasArgumentationScheme(SS_r, as_r)$, then $Argument(arg_s)$ can be a *critical question* attack. In the case of critical questions of the type *presumption* the locution has the semantics specified next:

  pre : $Agent(a_s)$ wants to attack the support $Argument(arg_r)$ of an $Agent(a_r)$ with an $Argument(arg_s)$, such that $hasSupportSet(arg, SS_s) \wedge Premise(pre_s) \wedge hasPresumption(SS_s, pre_s) \wedge hasPresumption(as_r, pre_s)$. $(Done[assert(a_r, a_s, arg_r), ...]) \wedge (\exists arg_s)(arg_s \vdash^{-} arg_r) \wedge$ $(\nexists arg_{s'} \in CS_s)(arg_s \vdash^{-} arg_{s'})$

post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe its support *Argument(*arg_r*)* and it is committed to provide an $Argument(arg_{rr})$ to support it or else to withdrawing it. Also, $Argument(arg)$ is inserted into the commitment store CS_s of $Agent(a_s)$.
$(\Diamond K_r \neg B_s arg_r) \wedge (K_r B_s arg) \wedge ((Feasible[\exists arg_{rr} \vdash^+ arg_r),$
$assert(a_r, a_s, arg_{rr})]) \vee (Feasible[\nexists arg_{rr} \vdash^+ arg_r), retract(a_r, a_s, arg_r)])) \wedge$
$(arg \in CS_s)$

In the case of critical questions of the type *exception* the locution has the following semantics:

pre : $Agent(a_s)$ wants to attack the support $Argument(arg_r)$ of an $Agent(a_r)$ with an $Argument(arg_s)$, such that $hasSupportSet(arg_s, SS_s) \wedge$
$Premise(pre_s) \wedge hasException(SS_s, pre_s) \wedge hasException(as_r, pre_s)$.
$(Done[assert(a_r, a_s, arg_r), ...]) \wedge (\exists arg_s)(arg_s \vdash^- arg_r) \wedge$
$(\nexists arg_{s'} \in CS_s)(arg_s \vdash^- arg_{s'})$

post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe its support *Argument(*arg_r*)*. Also, $Argument(arg_s)$ is inserted into the commitment store CS_s of $Agent(a_s)$.
$(\Diamond K_r \neg B_s arg_r) \wedge (K_r B_s arg_s) \wedge (arg_s \in CS_s)$

Finally, $Argument(arg_r)$ can be an attack argument to the $Argument(arg_{s'})$ of $Agent(a_s)$ with a *distinguishing-premise* or a *counter-example* in its support set such that $hasSupportSet(arg_r, SS_r) \wedge ((Premise(pr_r) \wedge$
$hasDistinguishingPremise(SS_r, pr_r) \vee (Case(c_r) \wedge$
$hasCounterExample(SS_r, c_r)))$. Then, $Argument(arg_s)$ can be an attack argument that rebuts $Argument(arg_r)$ with other *counter-example* or *distinguishing-premise*.

pre : $Agent(a_s)$ wants to rebut the attack $Argument(arg_r)$ of an $Agent(a_r)$ with an $Argument(arg_s)$, such that $hasSupportSet(arg_s, SS_s) \wedge ((Case(c_s) \wedge$
$hasCounterExample(SS_s, c_s)) \vee ((Premise(pr_s) \wedge$
$hasDistinguishingPremise(SS_s, pr_s)))$.
$(Done[attack(a_r, a_s, arg_r), ...]) \wedge (\exists arg_s)(arg_s \vdash^- arg_r) \wedge$
$(\nexists arg_{s''} \in CS_s)(arg_s \vdash^- arg_{s''})$

post : $Agent(a_r)$ knows that $Agent(a_s)$ does not believe its attack $Argument(arg_r)$. Also, $Argument(arg_s)$ is inserted into the commitment store CS_s of $Agent(a_s)$.
$(\Diamond K_r \neg B_s arg_r) \wedge (K_r B_s arg_s) \wedge (arg_s \in CS_s)$

- **{*pre*}** $accept(a_s, a_r, \phi)$ **{*post*}**
  This locution has different semantics depending on the content of ϕ. On one hand, ϕ can be a position $Solution(p)$ proposed by $Agent(a_r)$.

  pre : $Agent(a_s)$ wants to accept a position $Solution(p)$ proposed by $Agent(a_r)$.
  $(K_s B_r p) \wedge (B_s p)$

  post : $Agent(a_r)$ knows that $Agent(a_s)$ has accepted its position. Also, this position is inserted into the commitment store CS_s of $Agent(a_s)$ (and replaces a previous position if any).
  $(\Diamond K_r B_s p) \wedge (p \in CS_s) \wedge (\nexists p_s \in CS_s)(p_s \neq p)$

  On the other hand, ϕ can be an argument $Argument(arg_r)$ proposed by $Agent(a_r)$.

  pre : $Agent(a_s)$ wants to accept the $Argument(arg_r)$ proposed by $Agent(a_r)$ and there is not any inconsistent argument in the commitment store CS_s of $Agent(a_s)$.
  $(K_s B_r arg_r) \wedge (B_s arg_r) \wedge (\nexists arg_s \in CS_s)(arg_r \vdash^{-} arg_s)$

  post : $Agent(a_r)$ knows that $Agent(a_s)$ has accepted its argument. Also, this argument is inserted into the commitment store CS_s of $Agent(a_s)$.
  $(\Diamond K_r B_s arg_j) \wedge (arg_r \in CS_s)$

- **{*pre*}** $retract(a_s, a_r, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ wants to withdraw ϕ, which is an $Argument(arg_s)$ from its commitment store CS_s and reports it to any agent of the $Group(g)$ that is engaged in a dialogue with it.
  $(\neg B_s arg_s) \wedge (C_g B_s arg_s)$

  post : Every agent of the $Group(g)$ knows that $Agent(a_s)$ no longer believes $Argument(arg_s)$ and it is deleted from its commitment store.
  $(\Diamond C_g \neg B_s arg_s) \wedge (arg_s \notin CS_s)$

- **{*pre*}** $accept(a_s, all, \phi)$ **{*post*}**

  pre : $Agent(a_s)$ wants to close the dialogue and report all agents of the $Group(g)$ engaged in it the final $Solution(p)$ agreed to apply to the $Problem(q)$ at hand.
  $(C_g q) \wedge (C_g p)$

post : All agents $Agent(ag)$ of $Group(g)$ know that $Agent(a_s)$ believes $Solution(p)$. Also, their commitment stores are deleted and the dialogue ends.
$(\Diamond C_g B_s p) \wedge (\forall ag \in g)(\nexists CS_{ag})$

The axiomatic semantics is typically used to provide preliminary specifications of communication languages, but the knowledge of only its properties is not sufficient to understand a language.Thus, next section complements this semantics with an additional form of semantics that provide meaning for the transitions between the stages of the dialogue.

4.4.2 Operational Semantics

The operational semantics views the dialogue game protocol as an abstract state machine and defines precisely 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 4.3 shows the decision mechanisms that agents can use in each substage of the argumentation stage of our protocol. For clarity purposes, arrows labelled with the decision mechanism *D8* (presented below) from substages *R5*, *R6*, *R7* and *R8* to substage *R2* are omitted in the figure.

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,

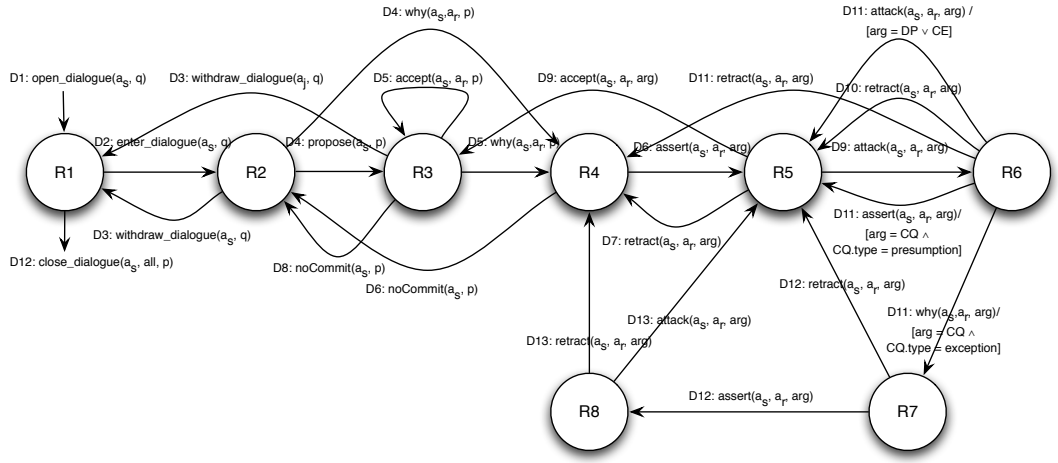

Figure 4.3: Decision Mechanisms of the Dialogue Game

if no specific output is invoked we denote this by a period in the third parameter of the triple $(\langle a_i, K, . \rangle)$.

Concretely, 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, uttering the locution *open_dialogue*(a_s, q) or not. The output of this mechanism is: *send(open_dialogue(a_s, ϕ))*.

- **D2 Enter or Close Dialogue:** A mechanism that allows an agent to decide to engage in a dialogue and utter the locution *enter_dialogue*(a_s, q) or not. 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 it enters in the dialogue or not. 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, ϕ))*, *listen()* or *send(close_dialogue(a_s, all, ϕ))*.

- **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 has not 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, ϕ))*.

- **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 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, ϕ))*, *send(why(a_s, a_r, ϕ))* 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 other 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, ϕ))* or *send(why(a_s, a_r, ϕ))*.

- **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, ϕ))*, *send(noCommit(a_s, ϕ))* 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, ϕ))*.

- **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, ϕ))*.

- **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. Else, 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, ϕ))* or *send(attack(a_s, a_r, ϕ))*.

- **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, ϕ))* 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, ϕ))*, *send(why(a_s, a_r, ϕ))*, *send(attack(a_s, a_r, ϕ))* or *send(retract(a_s, a_r, ϕ))*.

- **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 an 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, ϕ))*, *send(retract(a_s,*

a_r, ϕ)) 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 such information, it can counter-attack uttering the locution $attack(a_s, a_r, \phi)$. Otherwise, the agent has to withdraw its argument uttering the locution $retract(a_s, a_r, \phi)$. Thus, the outcomes of the mechanism are: *send(attack(a_s, a_r, ϕ))* or *send(retract(a_s, a_r, ϕ))*.

Now, we define 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:** $\langle a_s, D6, send(assert(a_s, a_r, \phi))\rangle \xrightarrow{L7} \langle a_s, D8, .\rangle$
- **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:** $\langle a_s, D13, send(retract(a_s, a_r, \phi))\rangle \xrightarrow{L10} \langle a_r, D6, listen()\rangle$

These transition rules provides the operational semantics of the dialogue, defining which is the range of potential decisions that agents can make in each stage of the dialogue.

4.5 Conclusions

This chapter has presented the dialogue game protocol that agents of our framework can use to interact and engage in argumentation dialogues. First, 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.

Then, we present our concept of dialogue strategies as a combination of the agents' profile and tactics. These strategies will select the best locution and content to put forward in each step of the argumentation dialogue. The approach presented here is based on the work presented in [Amgoud and Parsons, 2001][Amgoud and Hameurlain, 2006]. The latter work bases the selection of next moves in the dialogue on the agent's *strategic goals* and the selection of the content on the agent's *functional goals*. In our framework, the agent's strategic goals are represented by the knowledge stored in the argument-cases case-base and the agent's functional goals by the knowledge stored in the domain-cases case-base.

Finally, the *axiomatic* semantics and the *operational* semantics of the locutions are defined. The former specifies the pre-conditions that should be met to put forward each locution (or set of locutions) and the post-conditions that apply before their utterance. The latter views each locution as a transition in an abstract state-machine that represents the possible stages that can be reached during the dialogue.

Constructor Name	Syntax	Semantics
atomic concept **A**	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
datatypes **D**	D	$D^D \subseteq \Delta_D^{\mathcal{I}}$
abstrac role $\mathbf{R}_A$	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
datatype role $\mathbf{R}_D$	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_D^{\mathcal{I}}$
individuals **I**	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
data values	v	$v^{\mathcal{I}} = v^D$
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	$\exists U.D$	$(\exists U.D)^{\mathcal{I}} = \{x \mid \exists y.\, \langle x, y\rangle \in U^{\mathcal{I}} \text{ and } y \in D^D\}$
datatype value	$\forall U.D$	$(\forall U.D)^{\mathcal{I}} = \{x \mid \forall y.\, \langle x, y\rangle \in U^{\mathcal{I}} \rightarrow y \in D^D\}$
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[4]	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
individual equality	$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
concept existence	$\exists C$	$\#(C^{\mathcal{I}}) \geq 1$

Table 4.1: Syntax and Semantics of $\mathcal{SHOIN}(D)$ [Horrocks and Patel-Schneider, 2004].

	ASP	APL	CSP	CPL	CDP	AAP	AP$\emptyset$	CP$\emptyset$
Agreeable	√	√						
Disagreeable	√							
Open-Minded					√	√		
Argumentative				√	√			
Elephant's Child			√	√	√			√

Table 4.2: Agents' Profiles

Part IV

Evaluation

Chapter

5 Application to Water Management

5.1 mWater: Water-rights Negotiation Prototype

In this chapter, we apply our case-based argumentation framework to analyse, design and solve a resource allocation problem. The application domain consists on a social network of agents that must reach an agreement over a water-right transfer in a water market scenario like the one proposed in the *mWater* prototype [Botti et al., 2009b; Botti et al., 2009a; Botti et al., 2010; Garrido et al., 2009]. As reported in [Giret et al., 2010], it has been said that clean 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. Already Spain suffers from severe water shortages [Honey-Roses, 2007; Panayotou, 2007]. During the last years, the dramatical change of the Spanish Water Law has given rise to many water problems. Spain needs to improve its water management to meet the needs of different types of users (e.g. farmers, cities and private companies)

and to deal with its important 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 organism that specifies the rights that a user has over the water of the basin (e.g. the maximum volume that he can spend, the price that he must pay for the water or the district where it is settled[1]). For instance, a particular water right could allow its holder to pump out up to 10 m^3 of water per day during the next cotton season.

The purpose of mWater is to test and prove that the paradigm of agreement technologies can be successfully used in the construction of a prototype that will address the Water Rights Transfer problem. This application is of strategic importance for the Spanish society and economy. mWater will provide an efficient allocation of water resources based on a system of voluntary trade in water, which brings potentially large benefits to all parties involved. One implication of these complex requirements is the need for flexible on-demand negotiation, initiation, co-ordination, and supervision of various activities represented either through persons, or non-human actors (i.e. agents and services).

mWater will be a virtual market base system, in which water right transfer agreements will be executed by autonomous normative entities. In this market based environment different autonomous entities, representing individual or group of farmers, industries, or other water user entities, will get in contact with water right holders that want to transfer their rights. They will be able to negotiate the terms and conditions of the transfer agreement following the National Hydrological Plan and Basin Hydrological Plan normative laws. At the same time, the Basin Administration entities will be represented in the mWater system as normative or referee entities that will assure the correct execution of the water balanced distribution and usage. The focus of mWater will be on developing a good water right market design that can take into account the dynamics inherent in the water sector in a real aquifer of *la Mancha Oriental* (Spain).

The water market in Spain is strongly regulated and imposes several constraints. Firstly, water rights can only be transferred between users of the same or higher preference order defined by the Basin Hydrological Plan. For example, irrigation rights can only be transferred for alternative irrigation or human water supply but not for industrial uses. Furthermore, in this scenario we focus just on trading with water rights only for irrigation use and do not consider water trading for industrial uses, aquiculture,

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

leisure or navigation. Secondly, non consumptive rights can only be transferred to other non consumptive uses. Finally, both parts of the transfer (the seller and the buyer) must have the concession of the water right in property, thus preventing non-holders from participating in the market. This means that in our scenario only users that have previous irrigation rights can participate in the market.

The right-holders that participate in the market are allowed by law to establish the economic compensation by means of a private agreement process, signing a formal contract that is used as an official record. This record is publicly available to the other members of the basin in case some of them want to allege its applicability. When this transfer is detrimental to a third party, it can complain against this transfer. In such a case, the administrative organisms of the basin study the effects of the transfer and decide whether it is finally applicable.

In this scenario, it is possible to consider both the seller and the buyer as 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 at a higher 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 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. Also, we can make differences among the water rights to be traded depending on the river section. For instance, upstream water can be more valuable than downstream as its quality is significantly better.

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

- Water User: a water right-holder of the basin.
- 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 in problems among the contracting parties and (possibly) third parties of a water-right transfer agreement.
- Market facilitator: a management entity for assuring the execution of the different activities in the market scenario.

The dependency relation over these roles in an agent society S_t are the following (see Figures 5.1 to 5.7 for a graphical representation):

- Water User $<^{S_t}_{charity}$ Water User; Water User $<^{S_t}_{charity}$ Buyer; Water User $<^{S_t}_{charity}$ Seller; Water User $<^{S_t}_{authorisation}$ Third Party; Water User $<^{S_t}_{power}$ Administrator; Water User $<^{S_t}_{power}$ Jury; Water User $<^{S_t}_{power}$ Market facilitator

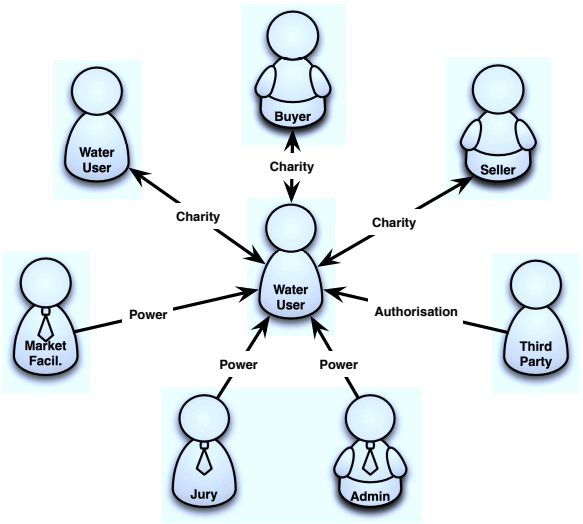

Figure 5.1: Water User Dependency Relations

- Buyer $<^{S_t}_{charity}$ Buyer; Buyer $<^{S_t}_{charity}$ Water User; Buyer $<^{S_t}_{charity}$ Seller; Buyer $<^{S_t}_{authorisation}$ Third Party; Buyer $<^{S_t}_{power}$ Administrator; Buyer $<^{S_t}_{power}$ Jury; Buyer $<^{S_t}_{power}$ Market facilitator
- Seller $<^{S_t}_{charity}$ Seller; Seller $<^{S_t}_{charity}$ Water User; Seller $<^{S_t}_{charity}$ Buyer; Seller $<^{S_t}_{authorisation}$ Third Party; Seller $<^{S_t}_{power}$ Administrator; Seller $<^{S_t}_{power}$ Jury; Seller $<^{S_t}_{power}$ Market facilitator

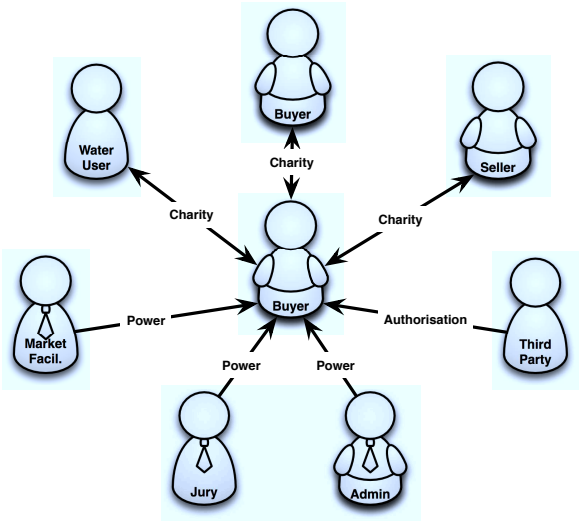

Figure 5.2: Buyer Dependency Relations

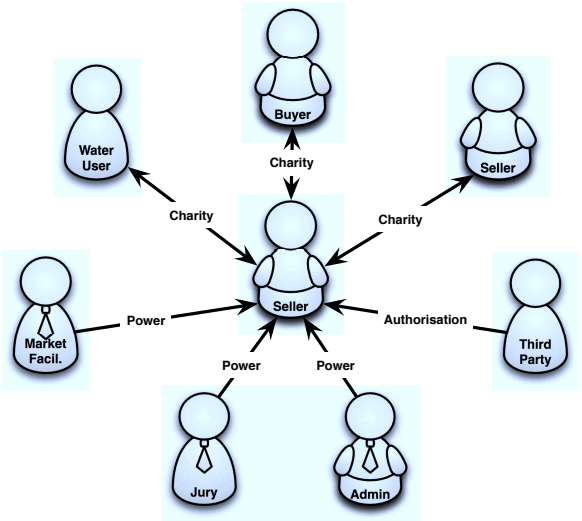

Figure 5.3: Seller Dependency Relations

- Third party $<^{S_t}_{charity}$ Third party; Third party $<^{S_t}_{power}$ Administrator; Third party $<^{S_t}_{power}$ Jury; Third party $<^{S_t}_{power}$ Market facilitator
- Administrator $<^{S_t}_{charity}$ Administrator; Administrator $<^{S_t}_{power}$ Jury; Administrator $<^{S_t}_{power}$ Market facilitator
- Jury $<^{S_t}_{charity}$ Jury; Jury $<^{S_t}_{power}$ Market facilitator

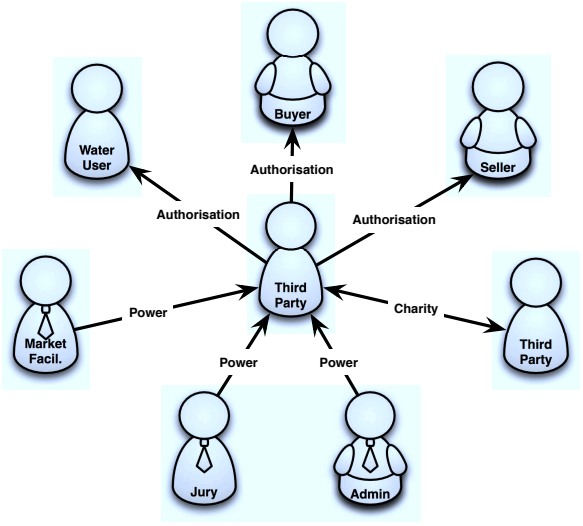

Figure 5.4: Third Party Dependency Relations

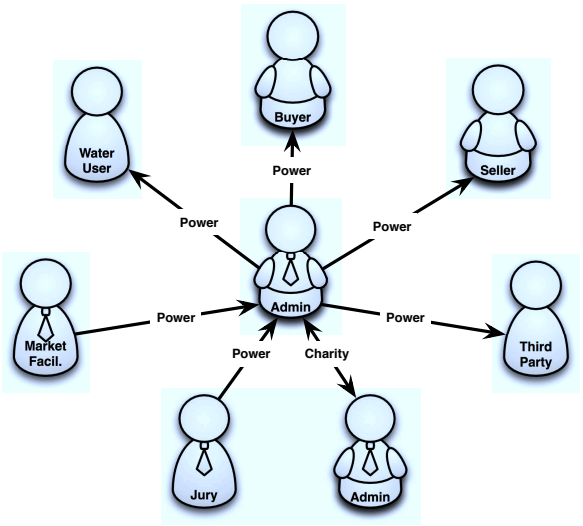

Figure 5.5: Administrator Dependency Relations

- Market facilitator $<^{S_t}_{charity}$ Market facilitator

Also, users and groups have their own normative context defined by a set of norms, a set of associated values (e.g. solidarity, economy, ecology) and a preference order over them. These are specified in the definition provided in next section for the example scenario.

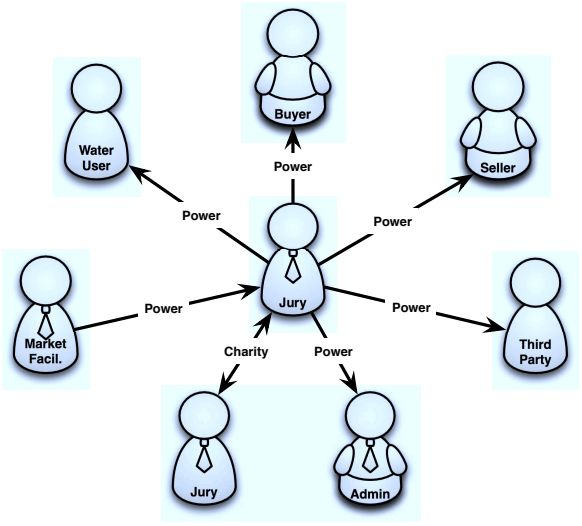

Figure 5.6: Jury Dependency Relations

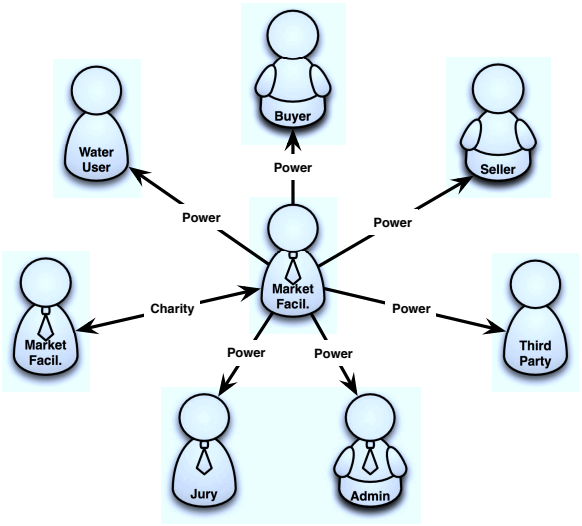

Figure 5.7: Market Facilitator Dependency Relations

This scenario has been entirely placed within a formal setting, where the operation of the system has the following phases:

1. A new trading table for the water market is opened by the market facilitator.
2. Users register in the market.
3. A new water-right transfer is offered to all users in the market.

4. Users argue to reach an agreement about the beneficiary of a water-right transfer by following a negotiation dialogue controlled by a dialogue game protocol. Here, third parties can allege the implications of the water-right assignment and a jury can take part if necessary.

5. A contract is signed.

6. An economic transaction is performed in the form of a compensation.

7. A record is publicly available in the given organisms to third parties.

By applying our argumentation framework in this system agents are able to reach agreements over water right transfers and decide the best transfer to perform according to different criteria. Next section introduces a particular setting in the mWater prototype according to a system that implements the argumentation framework proposed. Here, two farmers playing the role of buyers are arguing to be the beneficiary a water-right transfer with a basin administrator. Then, Section 5.3 shows an example where the abstract argumentation framework proposed in Chapter 3 is applied to this scenario. After that, Section 5.4 illustrates the reasoning process to generate, select and evaluate positions in this setting. Therefore, Sections 5.3 and 5.4 makes use of the example proposed in Section 5.2 to demonstrate the viability and use of the framework and the underlying reasoning process. Finally, Section 5.5 provides a discussion for the contributions of this chapter.

5.2 Example Scenario

Let us propose an example scenario in the mWater prototype, 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 5.8 shows a graphical representation of this scenario.

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

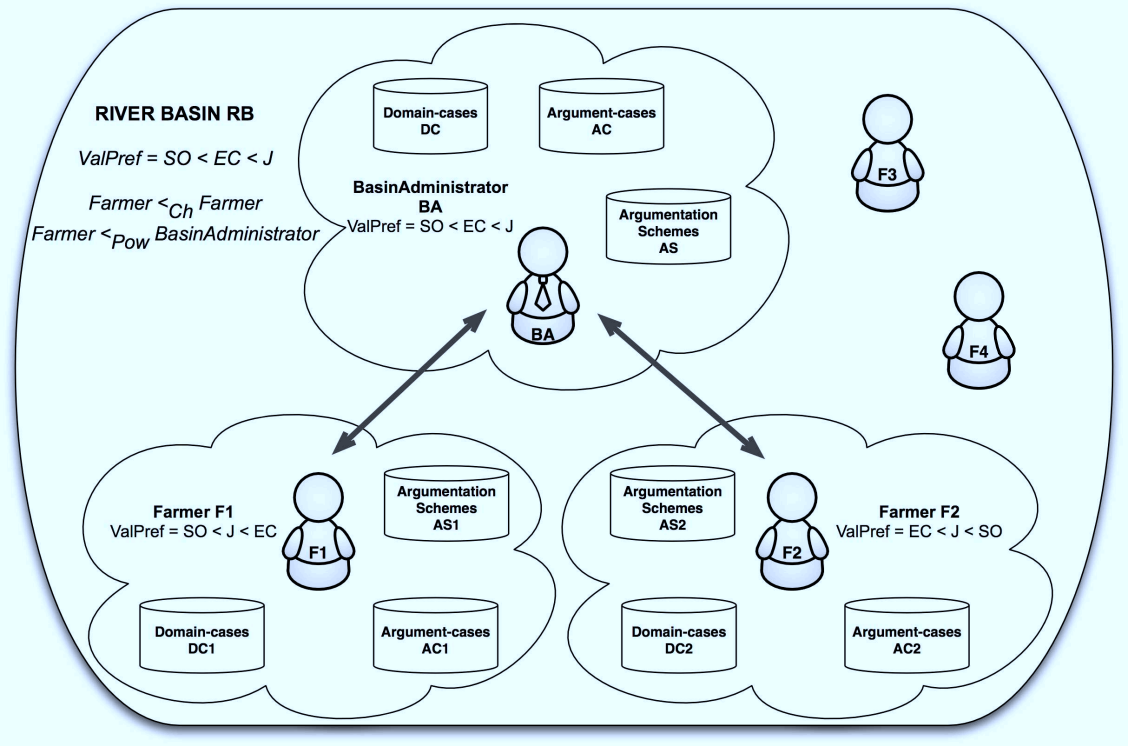

Figure 5.8: Water Market Scenario

BasinAdministrator). In addition, farmers prefer to reach an agreement before taking legal action to avoid the intervention of a jury (J). Also, $F1$ prefers economy (EC) over the intervention of a jury and this over solidarity (SO) ($SO <^{S_t}_{F1} J <^{S_t}_{F1} EC$), $F2$ prefers solidarity over the intervention of a jury and this over economy ($EC <^{S_t}_{F2} J <^{S_t}_{F2} SO$) and by default, BA has the value preference order of the basin, which promotes saving money in each transfer over being supportive with the personal needs of the basin users and tries to avoid the intervention of a jury in any case ($SO <^{S_t}_{BA} EC <^{S_t}_{BA} J$).

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

sources (domain-cases case-base, argument-cases case-base and argumentation schemes ontology).

5.3 Abstract Argumentation Framework

This section applies the abstract argumentation framework of Chapter 3 in the example of presented in the previous section. With this example, we illustrate the properties of the proposed argumentation framework for agent societies. In this scenario, $F1$ puts forward the argument:

> "I should be the beneficiary of the transfer because my land is adjacent to the owner's land".

Here, we suppose that the closer the lands the cheaper the transfers between them and then, this argument could promote economy. However, $F2$ replies with the argument:

> "I should be the beneficiary of the transfer because there is a drought and my land is almost dry".

In this argument, we assume that crops are lost in dry lands and helping people to avoid losing crops promotes solidarity. Also, agents know that if no agreement is reached, a jury must have to interfere and they want to avoid it. Then, they can also put forward the following arguments:

> "$F2$ should allow me ($F1$) to be the beneficiary of the water-right transfer to avoid the intervention of a jury (J)"

and

> "$F1$ should allow me ($F2$) to be the beneficiary of the water-right transfer to avoid the intervention of a jury (J)".

In addition, the BA knows that the jury will interfere if the agreement violates the value preferences of the river basin (which promotes solidarity over economy) and puts forward the argument:

> "$F2$ should allow $F1$ to be the beneficiary of the water-right transfer to avoid the intervention of a jury (J)".

In view of this context, the BA could generate an argumentation framework for an agent society defined as:

$$AFAS = <A, R, S_t>$$

Thus, we have the following arguments, which are all possible solutions for the water-right transfer agreement process:

A1 (posed by $F1$): $F1$ should be the beneficiary of the water transfer (denoted by $F1w$) to promote economy (EC).

A2 (posed by $F2$): $F1$ should not be the beneficiary of the water transfer (denoted by $F1nw$) to promote solidarity (SO).

A3 (posed by $F2$): $F2$ should be the beneficiary of the water transfer (denoted by $F2w$) to promote solidarity (SO).

A4 (posed by $F1$): $F2$ should not be the beneficiary of the water transfer (denoted by $F2nw$) to promote saving (EC).

A5 (posed by $F1$ and BA): $F2$ should allow $F1$ to be the beneficiary of the water transfer ($F1w\&F2nw$) to avoid the intervention of a Jury (J).

A6 (posed by $F2$): $F1$ should allow $F2$ to be the beneficiary of the water transfer ($F1nw\&F2w$) to avoid the intervention of a Jury (J).

The water transfer cannot be decided in favour of both water users, so *attacks(A1, A3)* and vice versa and we assume that a decision favouring at least one part must be taken, so *attacks(A2, A4)* and vice versa. In addition, a water user cannot be the beneficiary and not be the beneficiary of a water transfer at the same time, so *attacks(A1,A2)* and *attacks(A3,A4)* and vice versa. Also, *attacks(A5, A2)*, *attacks(A5, A3)* and *attacks(A5, A6)* and all these arguments attack A5 and *attacks(A6, A1)*, *attacks(A6, A4)* and *attacks(A6, A5)* and all these arguments attack A6. Then:

A=$\{$*A1, A2, A3, A4, A5, A6*$\}$

R =$\{$*attacks(A1, A3), attacks(A3, A1), attacks(A2, A4), attacks(A4, A2), attacks(A1, A2), attacks(A2, A1), attacks(A3, A4), attacks(A4,A3), attacks(A5, A2), attacks(A5, A3), attacks(A5, A6), attacks(A2, A5), attacks(A3, A5), attacks(A6, A5), attacks(A6, A1), attacks(A6, A4), attacks(A1, A6), attacks(A4, A6)*$\}$

S_t = $<$ *Ag, Rl, D, G, N, V, Role,* $Dependency_{S_t}$*, Group, Values,* $Valpref_Q$ $>$ *where:*

- $Ag = \{F1, F2, BA\}$
- $Rl = \{Farmer, BasinAdministrator\}$
- $D = \{Power, Charity\}$
- $G = \{RB\}$
- $N = N_{RB}$
- $V = \{EC, SO, J\}$
- *Role(F1) = Role(F2) = Farmer and Role(BA) = BasinAdministrator*
- $Dependency_{S_t} = \{Farmer <^{S_t}_{Pow} BasinAdministrator, Farmer <^{S_t}_{Ch} Farmer\}$
- *Group(F1) = Group(F2) = Group(BA) = RB*
- *Values(F1) = Values(F2) = Values(BA)* $= \{EC, SO, J\}$
- $Valpref_{F1} = \{SO <^{S_t}_{F1} J <^{S_t}_{F1} EC\}$, $Valpref_{F2} = \{EC <^{S_t}_{F2} J <^{S_t}_{F2} SO\}$, $Valpref_{BA} = \{SO <^{S_t}_{BA} EC <^{S_t}_{BA} J\}$

Therefore, the $AFAS$ for this example is shown in Figure 5.9.

Now, let us consider what happens with specific agents by creating their $AAFAS$. For instance, recalling that $F1$ prefers economy to other values and gives solidarity the lesser value ($SO <^{S_t}_{F1} J <^{S_t}_{F1} EC$) we have that:

$AAFAS_{F1}$ = $<$ *Ag, Rl, D, G, N, A, R, V, Role,* $Dependency_{S_t}$*, Group, Values, val,* $Valpref_{F1}$ $>$

Then, eliminating the unsuccessful attacks (due to value preferences of $F1$ and the power dependency relations) we have the equivalent $AFAS_{F1}$ for $AAFAS_{F1}$ as:

$AFAS_{F1}$ = $<$ A, $\{$*attacks(A1, A3), attacks(A1, A2), attacks(A1, A6), attacks(A4, A2), attacks(A4, A3), attacks(A4, A6), attacks(A5, A2), attacks(A5, A3), attacks(A5, A6)*$\}$, S_t $>$

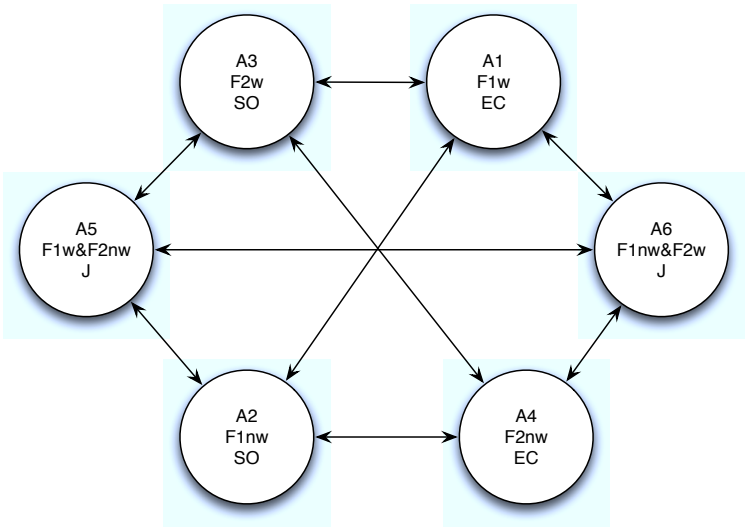

Figure 5.9: $AFAS$ Example

which is shown in the graph of Figure 5.10.

Now, we can compute the preferred extension(s) [Dung, 1995] of $AFAS_{F1}$ by taking into account the value preferences of $F1$ and the dependency relationships of the river basin. Then, from the $F1$ point of view, the $AFAS$ has the following preferred extension:

$$PE_{F1} = \{A1, A4, A5\}$$

meaning that $F1$ should be the beneficiary of the water-right transfer to promote economy and the no intervention of a jury.

In its turn, $F2$ gives the highest value to solidarity, but prefers to avoid a jury over economy ($EC <^{S_t}_{F2} J <^{S_t}_{F2} SO$). Therefore,

$$AAFAS_{F2} = < Ag,\ Rl,\ D,\ G,\ N,\ A,\ R,\ V,\ Role,\ Dependency_{S_t},\ Group,\ Values,\ val,\ Valpref_{F2} >.$$

Then, eliminating the unsuccessful attacks we have the equivalent $AFAS_{F2}$ for $AAFAS_{F2}$ as:

$$AFAS_{F2} = < A,\ \{attacks(A2,\ A1),\ attacks(A2,\ A4),\ attacks(A2,\ A5),\ attacks(A3,\ A1),\ attacks(A3,\ A4),\ attacks(A3,\ A5),\ attacks(A5,\ A2),\ at-$$

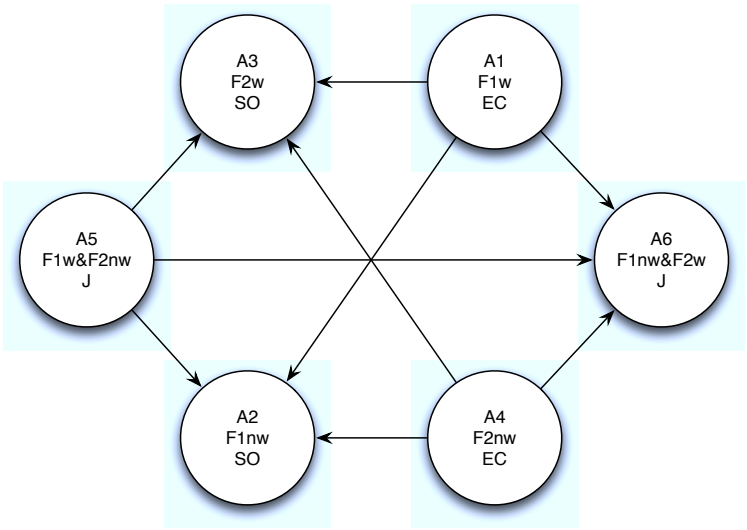

Figure 5.10: $AFAS_{F_1}$ Example

tacks(A5, A3), attacks(A5, A6), attacks(A6, A1), attacks(A6, A4)}, S_t >

which is shown in the graph of Figure 5.11.

For this case, the preferred extension would be:

$$PE_{F2} = A1, A4, A5$$

which means again that $F1$ should be the beneficiary of the water transfer. This demonstrates how the power dependency relation of BA prevails over farmers and their arguments. Otherwise, if we change the environment and set a charity dependency relation of basin administrators over farmers $Farmer <^{S_t}_{Ch} BasinAdministrator$, the preferences of $F2$ would prevail and the graph would be as the one of Figure 5.12.

In this case, the preferred extension would be:

$$PE_{F2}modified = \{A2, A3, A6\}$$

that would defend $F2$ as the beneficiary of the transfer agreement. Then, we have analysed under preferred semantics the strengths of the arguments submitted in the argumentation framework. This shows how each agent has its own view about the

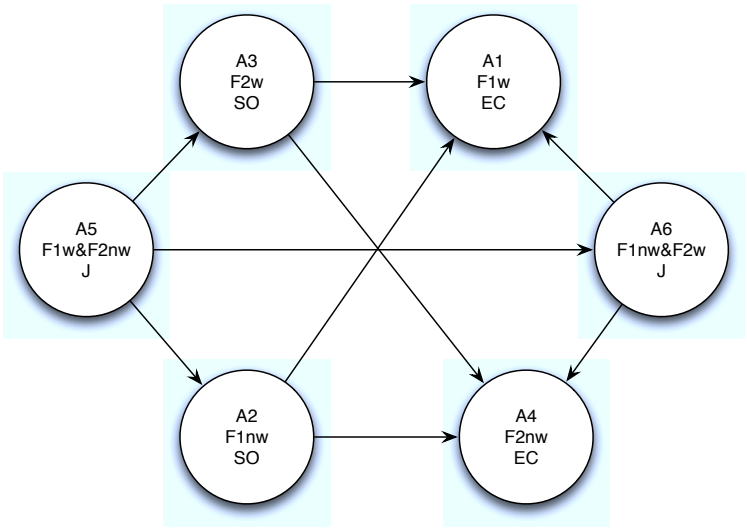

Figure 5.11: *AFAS F*2 Example

winning arguments (due to its value preferences) but, at the end, the dependency relations of the framework prevail and the basin administrator's arguments defeat the farmers'.

5.4 Reasoning Process

Once the abstract argumentation framework for the mWater proposed scenario has been proposed, we instantiate it by defining a particular structure for the arguments, based in the knowledge resources proposed in Chapter 3. Then, the reasoning process to generate, select and evaluate arguments is shown. During the process, agents use their own knowledge resources to manage arguments and follow the dialogue game protocol of Chapter 4 to interact.

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, the premises of the original problem could be as shown in Figure 5.13 and represent the identifier of the water right owner (*owner*), the offered volume in liters of water (*volume*), the price in Euros per liter of water (*price*), the district where the water right is settled (*district*) and the area of this district in acres (*area*).

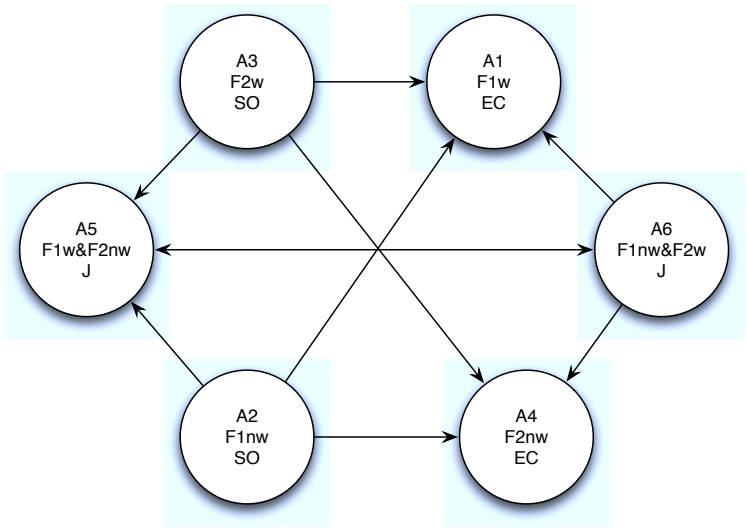

Figure 5.12: $AFAS_{F2}$ Modified Example

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, it enters in the dialogue by putting forward the locution *enter_dialogue(BA, q)*.

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 their case-bases of domain-cases ($DC1$ 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 in the left side of Figure 5.13. In this case, the solution consists of the identifier of the water-right transfer beneficiary (*beneficiary*) and the district of his land where the water has to be transferred (*tr district*). Figure 5.13 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 (was closer than 100 meters) to the land where the water-right was offered. Therefore, $F1$ can generate position pos_{F1} that is on the side of $F1$[2] and report this to the other

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

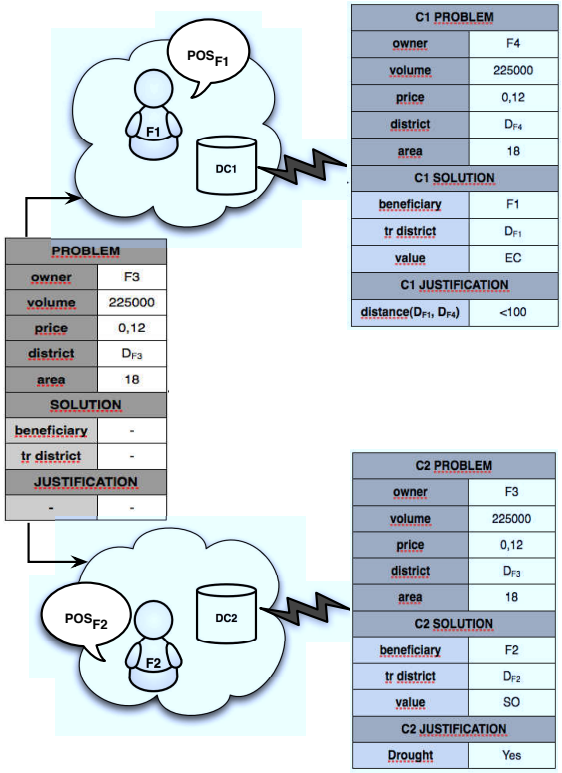

Figure 5.13: Generation of Positions

participants of the dialogue with the locution *propose(F1, pos_{F1})*.

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

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 for supporting 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 (according with the structure proposed in Chapter 3):

Support argument of $F1$ (with the locution *assert(F1, BA, SAF1)*):

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

Support 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 $F2tr$ means that the transfer is granted to $F1$ and $F2$ respectively. According with the values of the agents, we suppose that the closer the lands the cheaper the transfers between them and then, $SAF1$ would promote economy. Also, we assume that crops are lost in dry lands and 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, suppose that as basin administrator it knows an extra premise that states that there is a drought in the basin. First, this new premise matches an argumentation scheme of its ontology, $S1$, which changes the value preference order of the basin in case of drought (inspired in the 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 $EC <^{S_t}_{RB} J <^{S_t}_{RB} SO$ 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 $EC <^{S_t}_{RB} J <^{S_t}_{RB} SO$ in the case of drought.

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

Thus, the BA will check its case-base of domain-cases (DC) to find counter-examples for $C1$ and $C2$. As shown in Figure 5.14, 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}>}

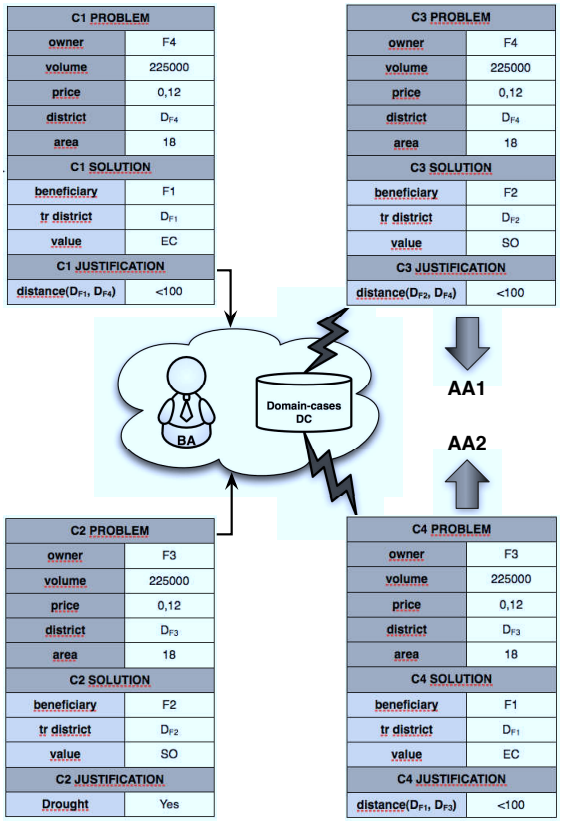

Figure 5.14: Counter-examples for $C1$ and $C2$

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

to $F1$:

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

In this case, $AA3$ undercuts $SAF1$ by attacking its support element $C1$ with the distinguishing premise *drought*. Again, we assume that attacking the argument of $F1$, the BA supports the argument of $F2$ and then promotes solidarity (SO).

Now, the BA has to select the argument that it will pose to attack the positions of the farmers. Note that, if we assume that agents always observe their value preference orders to put forward arguments, 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, it has still to decide which $AA1$ or $AA3$ would select to attack $SAF1$. To do that, it generates an argument-case for each argument and checks its argument-cases case-base to decide which is the best argument to pose in view of the previous experience. Now, let us suppose that the BA finds a similar argument-case for $AA3$ that was unaccepted at the end of the dialogue, shown in Table 5.1. However, the information of the group that the agents belong does not match with the current one. Therefore, the BA can infer that in the argument represented by this argument-case the agents belonged to a different river basin where solidarity is not promoted in case of drought. Finally, the BA finds a similar argument-case for $AA1$ that was accepted in the past. In this case, the social context and the value promoted match the current one. Thus, the BA will pose $AA1$ to attack the position of $F1$ and put forward the locution *attack(BA, F1, AA1)*. Note that if the social context of the argument-case retrieved for $AA3$ would have matched the current social context, the basin administrator would have a powerful reason to propose $AA1$ to attack $SAF1$. Also, it 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 any farmer ($Farmer <^{S_t}_{Power} Basin\ Administrator$). Then, it would try to generate more support for its position. In case that it cannot find such support, the F1 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.

PROBLEM	Domain Context	Premises = {owner=F3, volume=225000, ... drought=yes}	
	Social Context	Proponent	ID = BA
			Role = Basin Administrator
			Norms = N_{BA}
			ValPref = $EC <^{S_t}_{BA} J <^{S_t}_{BA} SO$
		Opponent	ID = F1
			Role = Farmer
			Norms = N_{F1}
			ValPref = $SO <^{S_t}_{F1} J <^{S_t}_{F1} EC$
		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 = {C5, C7}		
	Argumentation Schemes = $\emptyset$		
	Associated Dialogue Graphs		

Table 5.1: Argument-case retrieved for AA3

5.5 Discussion

With the scenario presented in this chapter we have demonstrated how agents' arguments can be computationally managed in the proposed argumentation framework. First, the abstract framework for the example scenario is presented and its properties analysed. From this analysis, we can compute the preferred extensions of the framework from the point of view of each agent, taking into account their preferences and the dependency relations of the society.

Then, the example shows the way in which agents can use the knowledge resources of the framework to generate, select and evaluate positions and arguments. Also, it takes into account the social context of agents to perform these activities and meets the requirements identified in Chapter 3.

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

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

For simplicity of the example proposed in this chapter, we have assumed that a proponent agent addresses its arguments to an opponent of its same group, having complete knowledge of the social context. However, either the proponent or the opponent's features could represent information about agents that act as representatives of a group and any agent can belong to different groups at the same time. In addition, the argumentation dialogue is centralised by the basin administrator and agents do not speak to each other directly, although the basin administrator could use the information provided by an agent to attack the arguments of other agent. However, our framework is conceived to serve both for mediated and for face-to-face argumentation dialogues.

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

Chapter

6 Application to Customer Support

6.1 Call Center Study Case

In this chapter, we evaluate the case-based argumentation framework presented in this thesis by running a set of empirical tests. With this objective, the framework has been implemented in the domain of a customer support application. Concretely, we consider a society of agents that act in behalf of a group of technicians that must solve problems in a Technology Management Centre (TMC). TMCs are entities which control every process implicated in the provision of technological and customer support services to private or public organisations. Usually, TMCs are managed by a private company that communicates with its customers via a call centre. This kind of centres allow customers to obtain general information, purchase products or lodge a complaint. They can also efficiently communicate public administrations with citizens. In a call centre, there are a number of technicians attending to a big amount of calls with different objectives –sales, marketing, customer service, technical support and any business or administration activity–. The call centre technicians have computers provided with a helpdesk software and phone terminals connected to a telephone switchboard that manages and balances the calls among technicians. The current implementation is

based in previous work that deployed a case-based multi-agent system in a real TCM [Heras et al., 2009c]. This system was implemented and is currently used by the TCM company. In the original implementation, agents were allowed to use their case-bases to provide experience-based customer support. In this thesis, the system has been enhanced by allowing agents to argue about the best way of solving the incidences that the call centre receives.

Nowadays to differentiate a company over other companies competing in the same market is very difficult. Products, prices and quality are very similar and companies try to obtain an advantage over their competitors by offering a careful attention to their customers. Most commercial activity is done via phone and it is necessary to avoid non-answered calls, busy lines, to ask the customer for repeating the query several times or to give incoherent answers. Moreover, a good customer support depends, in many cases, on the experience and skills of its technicians. A quick and accurate response to the customers problems ensures their satisfaction and a good reputation for the company and, therefore, it can increase its profits. Also, less experienced technicians are cheaper for the company. Thus, it is interesting to provide them with means to argue and solve (collaboratively if necessary) as many requests as possible. To store, and reuse later, the solution applied to each problem and the information about the problem-solving process could be a suitable way to improve the customer support offered by a company.

In a TMC, there are a number of technicians whose role is to provide the customers with technical assistance –microcomputing, security and network management among other services–. This help is typically offered via a call centre. Usually, the staff of a call centre is divided into three levels:

- First level operators (or *Operators*), who receive customer queries and answer those ones from which they have background training or their solution is registered in the company manuals of action protocols.
- Second level operators (or *Experts*), who are expert technicians that have more specialised knowledge than the first level operators and are able to solve problems that the operators cannot solve.
- *Administrators*, who are in charge of organising working groups, of assigning problems to specific operators and of creating generic solutions, which will be registered and used later by the operators of lower levels.

Therefore, we consider a society S_t composed by call-centre technicians with three

possible roles: operator, expert and administrator. The defined dependency relations are the following:

- Operator $<^{S_t}_{charity}$ Operator; Operator $<^{S_t}_{authorisation}$ Expert; Operator $<^{S_t}_{power}$ Administrator
- Expert $<^{S_t}_{charity}$ Expert; Expert $<^{S_t}_{power}$ Administrator
- Administrator $<^{S_t}_{charity}$ Administrator

Also, each technician can have his own values (e.g. efficiency, accuracy, savings), his own preferences over them and belong to different groups intended to solve specific types of problems or assigned to specific projects. Also, these groups can have their own social values.

To guarantee a high-quality service, the company subscribes to a Service Level Agreements (SLAs) with the customer, where the different characteristics of the services to provide are specified –service descriptive labels that identify each request as belonging to a certain type (category trees), service priority, attention and assistance maximum times and certain parameters that measure the fulfilment degree of these services–. In case of breach of the agreements, the company is economically penalised.

Once the SLAs have been established, the customer can request the supply of the services that have been agreed by means of several entry channels –phone call, website, e-mail, post and fax–. When the centre receives the request, the so-called incidence register or *ticket* is generated with the customer data and a description of the incidence. Hence, this ticket is the problem to be solved. From the customer point of view, the tickets are fundamentally characterised by their state –assigned, in progress, solved, closed, pending, require external provider or require software development– and by the problem-solving time, which allows the customer to know the degree of the agreements fulfilment. For the centre, the tickets are also characterised by other parameters, such as the type or category of the incidence (e.g. network error, OS exception, hardware failure, etc.), data about the actual problem occurred inside each category (e.g. OS, hardware brand, customer observations, etc.), to which group the ticket has been assigned or work-notes about the incidence. The problem-solving process generates more information that helps to explain the solution that has been applied–solving-method, operator level, keywords, URL's, attached documents or observations–. Table 6.1 shows the current attributes that have been used to characterise tickets in the tests performed in this section.

Parameter	Definition
Ticket ID	Ticket identifier.
Category	Incidence type.
Attributes	Data that defines the specific incidence. The number and type of attributes depends on the category.
Problem description	Textual description about the incidence.
Project	Project or contract that binds the company running the call center to provide support to the customer that reports the incidence.
Solving groups	Group(s) of technicians that have attended the incidence.
Solving technicians	Technician(s) that have attended the incidence.
Solution ID	Identifier of the solution applied to solve the incidence.
Solution description	Textual description of the solution applied to the incidence.
Times used	Number of times that this solution has been applied to solve this type of incidence.
Promoted value	Value promoted with the solution applied to this incidence.

Table 6.1: Call Center Ticket Attributes

In this application domain we assume that each technician has a helpdesk application to manage the big amount of information that processes the call centre. The basic functions of this helpdesk are the following:

- To register the ticket information: customer data, entry channel and related project, which identifies the customer and the specific service that is being provided.
- To track each ticket and to scale it from one technician to a more specialised one or to a different technician in the same level.
- To warn when the maximum time to solve an incidence is about to expire.
- To provide a solution for the ticket. This means to generate an own position or to ask for help to the members of a group.

In addition, this helpdesk would implement an argumentation module to solve each ticket as proposed in our framework. Hence, we assume the complex case where a ticket must be solved by a group of agents representing technicians that argue to reach an agreement over the best solution to apply. Each agent has its own knowledge resources (acceded via his helpdesk) to generate a solution for the ticket. The data-flow for the problem-solving process (or argumentation process) to solve each ticket is the following:

1. The system presents a group of technicians with a new ticket to solve.
2. If possible, each technician generates his own position by using the argumentation module. This module supports the argumentation framework proposed in this thesis.

3. All technicians that are willing to participate in the argumentation process are aware of the positions proposed in each moment.

4. The technicians argue to reach an agreement over the most suitable solution by following a deliberation dialogue controlled by the proposed dialogue game protocol.

5. The best solution is proposed to the user and feedback is provided and registered by each technician helpdesk.

To date, the helpdesk of each technician is provided with a case-based reasoning engine that helps them to solve the ticket. The new argumentation module will allow different technicians to reach agreements over the best solution to apply in each specific situation. In this example application, we assume that the most efficient technicians are acknowledged and rewarded by the company. Therefore, each technician follows a *persuasion* dialogue with their partners, trying to convince them to accept its solution as the best way to solve the ticket received, while observing the common objective of providing the best solution for a ticket from its point of view.

6.2 Implementation Details

A system that provides support to the technicians of a call centre were implemented in a helpdesk application [Heras et al., 2009c]. In this thesis, this work has been extended by integrating an argumentation module that implements the case-based argumentation framework proposed. In the current system, the technicians are represented by agents that access an automated helpdesk and argue to solve an incidence. Every agent has individual CBR resources and preferences over values.

Also, each agent can have its own values (e.g. efficiency, accuracy, etc.), its own preferences over them and belong to different groups intended to solve specific types of problems or assigned to specific projects. These groups can have their own social values.

Following, we describe the different modules of the system and their functionality:

- **Magentix2**: to develop this system we have used the Magentix2 agent platform[1].

[1] http://users.dsic.upv.es/grupos/ia/sma/tools/magentix2/index.php

Magentix2 is an agent platform that provides new services and tools that allow for the secure and optimized management of open MAS.

- **Domain CBR module**: consists of a CBR module with data about previous problems solved in the call centre (domain-cases). This CBR is initialised with past tickets of the helpdesk application. The CBR module used to perform the tests is an improved version of the module used in [Heras et al., 2009c]. To make a query to the domain CBR, the user has to provide a ticket and a threshold of similarity. The domain CBR module searches the domain case-base and returns a list of similar domain-cases to the given ticket. In addition, with every request attended and every CBR cycle performed, the module adds, modifies or deletes one or more domain-cases of the domain case-base. In the current version of the system, if the ticket that has been solved is similar enough (over certain similarity threshold) to a case of the domain-cases case-base, the update algorithm updates this case with the new data acquired. Otherwise, a new domain-case is created and added to the case-base.

- **Argumentation CBR module**: consists of a CBR module with argumentation data (previous arguments stored in the form of argument-cases). Once an agent has a list of potential solutions for a current incidence, it has to select the best position to put forward among them. Also, the agent can generate arguments to support its position and attack other agent's arguments and positions. Then, this module is used to look for previous argumentation experiences and use this knowledge to select the best positions and arguments to propose. Thus, argument-cases store information related to the domain and the social context where previous arguments (and their associated positions) were used. The information about the domain consists of a set of features to compare cases (e.g. the type of incidence or the affected equipment) and information about the social context where the proposed solution was applied (e.g. the agents that participated in the dialogue to solve the problem, their roles or their value preferences). The latter information can determine if certain positions and arguments are more persuasive than others for a particular social context and hence, agents can select the best ones to propose in the current situation. As for the domain domain-cases case base, if the argument-cases created during the problem solving process are similar enough to previous argument-cases stored in the argument-cases case-base, the update algorithm updates those cases with the new data acquired. Otherwise, new argument-cases are created and added to the case-base.

- **Argument-cases Ontology**: consists in a OWL 2[2] ontology with the argumentation-schemes used to generate positions and arguments. In the current version of the system, the agents only use a simplified version of the *Argument from Expert Opinion* [Walton, 1996]. Among the critical questions of this argumentation-scheme, agents can use the *Consistency question*, in the case that the attacking agent is also an expert but has a different conclusion for the same argument or the *Backup Evidence question*, to ask the proponent of the argument for evidences that support its proposal. These attacks can be rebutted with distinguishing premises or counter-examples, which are shown as evidences that support the argument of the proponent agent.

- **Ontology parsers**: The contents of the case-bases of the domain CBR and the argumentation CBR are stored as objects in OWL 2 ontologies. In this way, heterogeneous agents can use these ontologies as common language to interchange solutions and arguments generated from the case-bases of the argumentation framework. The main advantage of using ontologies is that the structures and features of the cases are well specified and agents can easily understand them. The ontology parsers developed provide an API to read and write data in the case-bases of the argumentation module.

- **Argumentation agent**: is an agent with a domain CBR and an argumentation CBR capable to engage in an argumentation dialogue to solve an incidence. This agents learn about the domain problem and the argumentation dialogue adding and updating cases into the domain and argumentation case-bases with each CBR run. Moreover, the agent can play any role defined before. In our implementation, this agent is a extension of Magentix2 Base-Agent[3].

- **Commitment Store**: is a resource of the argumentation framework that stores all the information about the agents participating in the problem solving process, argumentation dialogues between them, positions and arguments. By making queries to this resource, every agent of the framework can read the information of the dialogues that it is involved in.

In order to show how the developed system works, the data-flow for the problem-solving process (or argumentation process) to solve each ticket is shown in Figure 6.1

[2] `http://www.w3.org/TR/owl2-overview/`

[3] http://users.dsic.upv.es/grupos/ia/sma/tools/magentix2/archivos/javadoc/-es/upv/dsic/gti_ia/core/BaseAgent.html

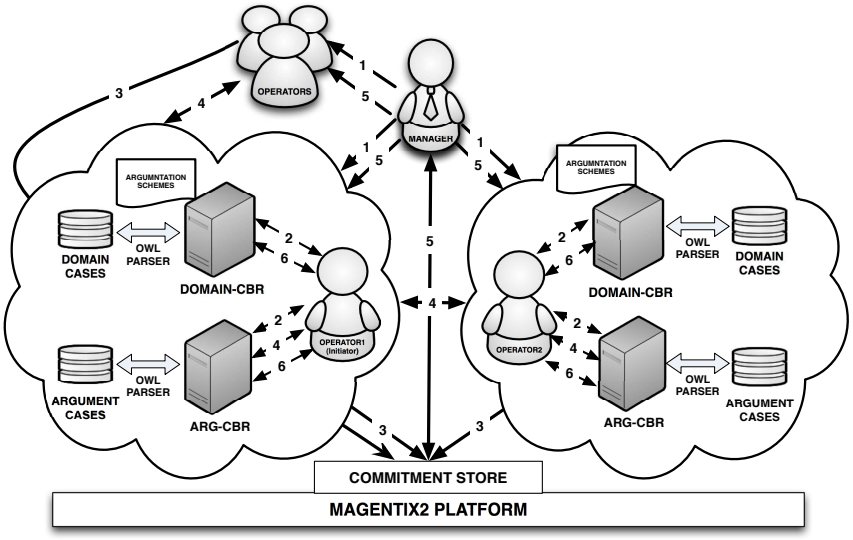

Figure 6.1: Data-flow for the argumentation process of the helpdesk application

and described below (arrows in the figure are labelled with the number of the data-flow step that they represent):

1. First, we have some argumentation agents running in the platform and representing the technicians of the call centre. An agent of the group (randomly selected) acts as the *initiator* of the argumentation dialogue. This kind of agent has a special behaviour to receive tickets to solve and create a new dialogue with the agents of its group. The process begins when a ticket that represents an incidence to solve is received by the initiator agent. Then, this agent sends the ticket to their partners in the group.

2. Each agent evaluates individually if it can engage in the dialogue offering a solution. To do that, the agent makes a query to its domain CBR to obtain potential solutions to the ticket based on previous solutions applied to similar tickets. To compute such similarity, agents use a *weighted Euclidean* algorithm that searches their domain-cases case-bases for previous problems that semantically match the category of the current ticket to solve. Thus, the algorithm retrieves all problems of the same category and of related categories and select those that syntactically match (assign the same values to the attributes that match the ticket attributes) and overpass a defined *similarity threshold*. If one or more valid solutions can be generated from the selected domain-cases, the agent will be able to defend a position in the dialogue. We consider a valid solution any domain case from

the domain CBR with one or more solutions and with a similarity degree greater than the given threshold. Moreover, the agent makes a query to its argumentation CBR for each possible position to defend. With these queries a *suitability degree* of the positions is obtained as explained in Chapter 3. This degree represents if a position will be easy to defend based on past similar argumentation experiences. Then, all possible positions to defend are ordered from less to more suitability degree.

3. When the agents have a position to defend (a proposed solution), these positions are stored by the commitment store, such that other agents can check the positions of all dialogue participants. Every agent tries to attack the positions that are different from its position.

4. The argumentation process consists on a series of steps by which agents try to defend its positions by generating counter-examples and distinguishing premises for the positions and arguments of other agents. A counter-example for a case is generated by retrieving from the domain case-base other case that matches the features of the former, but has a different conclusion. Similarly, distinguishing premises are computed by selecting such premises that the agent has taken into account to generate its positions, but that other agents did not considered. If different attacks can be generated, agents select the best attack to rebut the position of other agent by making a query to their argument-cases case-base, extending the characterisation of each case with the current social context. In this way, agents can gain knowledge about how each potential attack worked to rebut the position of an agent in a past argumentation experience with a similar social context. When an agent is able to rebut an attack, the opponent agent makes a vote for its position. Otherwise, the agent must withdraw its position and propose an alternative position, if possible.

5. The dialogue finishes when no new positions or arguments are generated after a specific time. The initiator agent is in charge of making queries to the commitment store agent to determine if the dialogue must finish. Then, this agent retrieves the active positions of the participating agents. If all agents agree, the solution associated to the agreed position is selected. Otherwise, the most frequent position wins. In case of draw, the most voted position in selected. If even in this case the draw persists, a random choice is made. Finally, the initiator agent communicates the final solution (the outcome of the agreement process) to

the participating agents.

6.3 Evaluation Criteria

To evaluate the proposed case-based argumentation framework, a set of empirical tests have been performed. For the tests, we assume that there are several agents engaged in an agreement process and that these agents have an individual argumentation system that complies with our case-based argumentation framework. Testing a CBR system involves two separated processes: verification (concerned with building the system right) and validation (concerned with building the right system) [Watson, 1997]. Validation is a complex socio-technical problem that involves to ensure that the developed system is the right system for the problem to solve. Here we cope with the verification problem and more concretely, with the problem of verifying the performance of the system. The set of variables that can be modified in the tests to verify the framework from different perspectives are shown in Table 6.2:

Variable Name	**Definition**
#Agents	Number of participating agents
Relations	Dependency relations among different roles (charity, authorisation or power)
Agent Profiles	Profile of the participating agents (agreeable, disagreeable, open-minded, argumentative or elephant's child)
Similarity Threshold	Threshold over which two cases are considered as similar
#Domain-cases	Number of cases in the domain-cases case-base
#Argument-cases	Number of cases in the argument-cases case-base
Similarity Degree Weight (w_{SimD})	The weight of the similarity degree to compute the suitability degree
Support Factor Weight (w_{SF})	The weight of the support factor to compute the suitability degree
Persuasion Degree Weight (w_{PD})	The weight of the persuasion degree to compute the suitability degree
Support Degree Weight (w_{SD})	The weight of the support degree to compute the suitability degree
Risk Degree Weight (w_{RD})	The weight of the risk degree to compute the suitability degree
Attack Degree Weight (w_{AD})	The weight of the attack degree to compute the suitability degree
Efficiency Degree Weight (w_{ED})	The weight of the efficiency degree to compute the suitability degree
Explanatory Power (w_{EP})	The weight of the explanatory power to compute the suitability degree

Table 6.2: Experimental Variables for the Tests

For the tests, a real database of 200 tickets solved in the past is used as domain knowledge. Translating these tickets to domain-cases, we have obtained a tickets case-base with 48 cases. Despite the small size of this case-base, we have rather preferred to use actual data instead of a larger case-base with simulated data. The argument-cases case-bases of each agent are initially empty and populated with cases as the agents acquire argumentation experience in execution of the system. To diminish the influence of random noise, for each round in each test, all results report the average

and confidence interval of 48 simulation runs at a confidence level of 95%, thus using a different ticket of the tickets case-base as the problem to solve in each run. The results report the mean of the sampling distribution (the population mean) by using the formula:

$$\mu = \bar{x} \pm t * \frac{s}{\sqrt{n}} \tag{6.1}$$

where, $\bar{x}$ is the sample mean (the mean of the 48 experiments), t is a parameter that increases or decreases the standard error of the sample mean ($\frac{s}{\sqrt{n}}$), s is the sample standard deviation and n is the number of experiments. For small samples, say below 100, t follows the *Student's t-distribution*, which specifies certain value for the t parameter to achieve a confidence level of 95% for different sizes of population. In our case, with a population of 48 experiments the Student's t-distribution establishes a value of 2.0106 for t.

In each simulation experiment, an agent is selected randomly as initiator of the discussion. This agent has the additional function of collecting data for analysis. However, from the argumentation perspective, its behaviour is exactly the same as the rest of agents and its positions and arguments do not have any preference over others (unless there is a dependency relation that states it). The initiator agent receives one problem to solve per run. Then, it contacts its partners (the agents of its group) to report them the problem to solve. If the agents do not reach an agreement after a maximum time, the initiator chooses the most supported (the most voted) solution as the final decision (or the most frequent in case of draw). If the draw persists, the initiator makes a random choice among the most frequent solutions.

6.4 Evaluation Tests

The case-based argumentation framework proposed in this thesis has been evaluated from different perspectives. On one hand, the *performance* of the system that implements the framework in the customer support application domain is tested and analysed. This systems consists of a module that allows agents that support our case-based argumentation framework to communicate and solve tickets together. On the other hand, we have run a set of tests intended to evaluate the *suitability* of each strategy proposed in Chapter 4. Finally, the ability of the system to take into account the *social context* of the participating agents is also verified.

6.4.1 Testing the Performance

The performance tests have been repeated and their results compared for the following decision policies:

- Random policy (CBR-R): each agent uses its Domain CBR module to propose a solution for the problem to solve. Then, a random choice among all solutions proposed by the agents is made. Agents do not have an Argumentation CBR module.
- Majority policy (CBR-M): each agent uses its Domain CBR module to propose a solution for the problem to solve. Then, the system selects the most frequently proposed solution. Agents do not have an Argumentation CBR module.
- Argumentation policy (CBR-ARG): agents have Domain and Argumentation CBR modules. Each agent uses its Domain CBR module to propose a solution for the problem to solve and its Argumentation CBR module to select the best positions and arguments to propose in view of its argumentation experience. Then, agents perform an argumentation dialogue to select the final solution to apply.

To evaluate the effect of the available argumentative knowledge that agents have, some tests are also repeated for the following specific settings of the argumentation policy. These settings cover the more interesting options regarding which agents have argumentation skills:

- CBR-ARG All-Argument-Cases (CBR-ARG AAC): All participating agent have argument-cases in their argument-cases case-base.
- CBR-ARG Initiator-Argument-Cases (CBR-ARG IAC): Only one agent, say the initiator agent, has argument-cases in its argument-cases case-base. Note that the selection of the initiator as the agent that has argumentative knowledge is just made for the sake of simplicity in the nomenclature. The behaviour of this agent only differs from the other agents' in the fact that it is in charge of starting the dialogue process and conveying the information about the final outcome. This do not affect its performance as dialogue participant and does not grant this agent any privileges over their partners.

- CBR-ARG Others-Argument-Cases (CBR-ARG OAC): All agents except from one, say the initiator, have argument-cases in their argument-cases case-bases.

With these tests, we evaluate the efficiency of the system that implements our framework under the different decision policies. By default, all agents know each other, all are in the same group and the dependency relation between them is *charity*. The values of each agent have been randomly assigned and agents know the values of their partners. Also, all agents play the role of *operator*. The influence of the social context will be evaluated in the Section 6.4.3. In addition, the agents that follow the argumentation policy assign weights to the *similarity degree* (w_{SimD}) and the *support factor* (w_{SF}) proportionally to the number of domain-cases and argument-cases that they have in their case-bases. Depending on the application domain, a different assignment for the weights could influence the performance of the system. However, due to the reduced size of our tickets case-bases, a proportional assignment is be suitable enough for the objectives of this performance evaluation. Equation 6.2 shows the simple rule that has been used in the tests.

$$\begin{aligned} w_{SimD} &= \frac{\#domaincases}{\#domaincases + \#argumentcases} \\ w_{SF} &= \frac{\#argumentcases}{\#domaincases + \#argumentcases} \end{aligned} \tag{6.2}$$

Also, by default agents do not follow any dialogue strategy, setting the same weight for all elements of the support factor.

6.4.1.1 Number of cases that the framework learns with respect of the time.

To perform this test, all agents follow the argumentation policy, with an initial number of 5 domain-cases in their domain-cases case-bases. The argument-cases case-base of all agents are initially empty. In each iteration, the agents use their CBR modules to propose and select positions and arguments and after this process, each agent updates its case-bases with the knowledge acquired.

If the system works properly, the knowledge acquired about past problem solving processes should increase with the time until some threshold, where the learning process should stabilize (because the cases in the case-bases of the agents cover most possible problems and arguments in the domain). To perform this test, we have executed several rounds to simulate the use of the system over certain period of time. For each repetition,

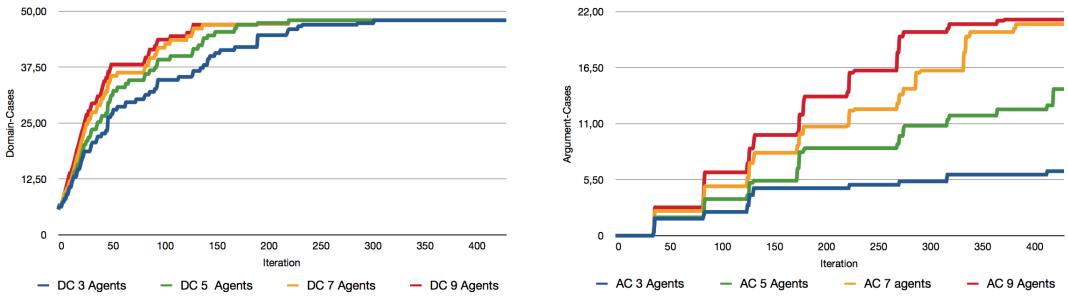

Figure 6.2: Number of domain-cases (left) and argument-cases (right) that agents learn.

we compute the average number of domain-cases and argument-cases in the case-bases of the agents. Figure 6.2 shows the results obtained in this test. The experiment has been repeated for 3, 5, 7 and 9 agents and the average number of domain-cases (DC) and argument-cases (AC) that all agents learn in each iteration has been computed. As expected, in all cases, the agents are able to learn the 48 domain-cases of the tickets case-base. However, if more agents participate in the dialogue, the quantity of domain knowledge that agents have available and interchange among them increases and the domain-cases case-bases are more quickly populated. Also, the quantity of argument-cases that agents are able to learn increases with the number of agents, since more potential positions and arguments give rise to more complex argumentation dialogues. As shown in the figure, the learning curve for the argument-cases is less soft than for the domain-cases, presenting peaks at some points. This is due to the fact that at some points of the dialogue, the agents can learn a specific domain-case that change its opinion about the best solution to apply for a specific category of problem. Therefore, the outcome of subsequent dialogues differ from the outcome that could be expected taking into account past similar dialogues and the argument-cases learning rate of the agents in those situations notably increases.

The results of this test have helped us to set the value of some parameters of the subsequent evaluation tests. The test shows that in 48 simulation runs, 3 agents are able to learn an average of the 54.8 % domain-cases of the tickets case-base, 5 agents the 56,6 %, 7 agents the 66,9 % and 9 agents the 73.1 %. The maximum number of argument-cases that agents are able to learn reaches an average of 20 argument-cases when 9 agents participate in the dialogue (18 argument-cases in the worst case). Therefore, due to the small size of the whole tickets case-base and the learning rates obtained in this test, the evaluation tests have been executed with a maximum number of 9 agents

participating in the dialogue, with domain-cases case-bases populated with a maximum number of 45 domain-cases and argument-cases case-bases populated with a maximum number of 20 argument-cases (except from the social context tests, where a more varied choice of social contexts enables the learning of a larger number of argument-cases). Thus, the domain-cases of the case-bases of the agents will be randomly populated and increased from 5 to 45 cases in each experimental round. The argument-cases case-bases of the agents for the argumentation-based policy are populated with 20 randomly selected argument-cases (from those acquired during the performance of the present test). Also, to evaluate the influence of the quantity of argumentative knowledge that the agents have in some tests, those tests are repeated for the case of 7 agents, setting the number of domain-cases of the case-bases of the agents to 20 (approximately the half part of the available cases in the tickets case-base), while varying the number or argument-cases of the argumentative agents from 0 to 18 cases, with an increase of 2 randomly selected argument-cases in each round. The number of the agents for these tests has been set to 7 to allow complex enough argumentation dialogues where the use of argument-cases can be useful, while having a reasonable case learning rate to avoid filling the case-bases with all the available knowledge for this case of study with a small number of simulations.

6.4.1.2 Percentage of problems that were solved with respect to the knowledge of the agents.

In this test, the percentage of problems that the system is able to solve (provide a solution, regardless of its level of suitability) has been tested under the different decision policies. The test has been repeated for different assignments of the domain and argumentative knowledge that agents have. One can expect that having more domain and argumentative knowledge (more domain and argumentation cases in the case-bases) will prevent less problems from being undecided.

As shown by Figure 6.3, in all cases all policies achieve the same results. As expected, the percentage of problems solved by the system increases with the number of domain-cases and argument-cases. In addition, the quantity of problems that remain undecided decreases as the number of participating agents increases. Obviously, with more agents in the system the probability that an agent has a suitable solution for the problem at hand in its case-bases increases. In addition, if the number of domain-cases is fixed and the quantity of argumentative knowledge of the agents that follow the argumentation

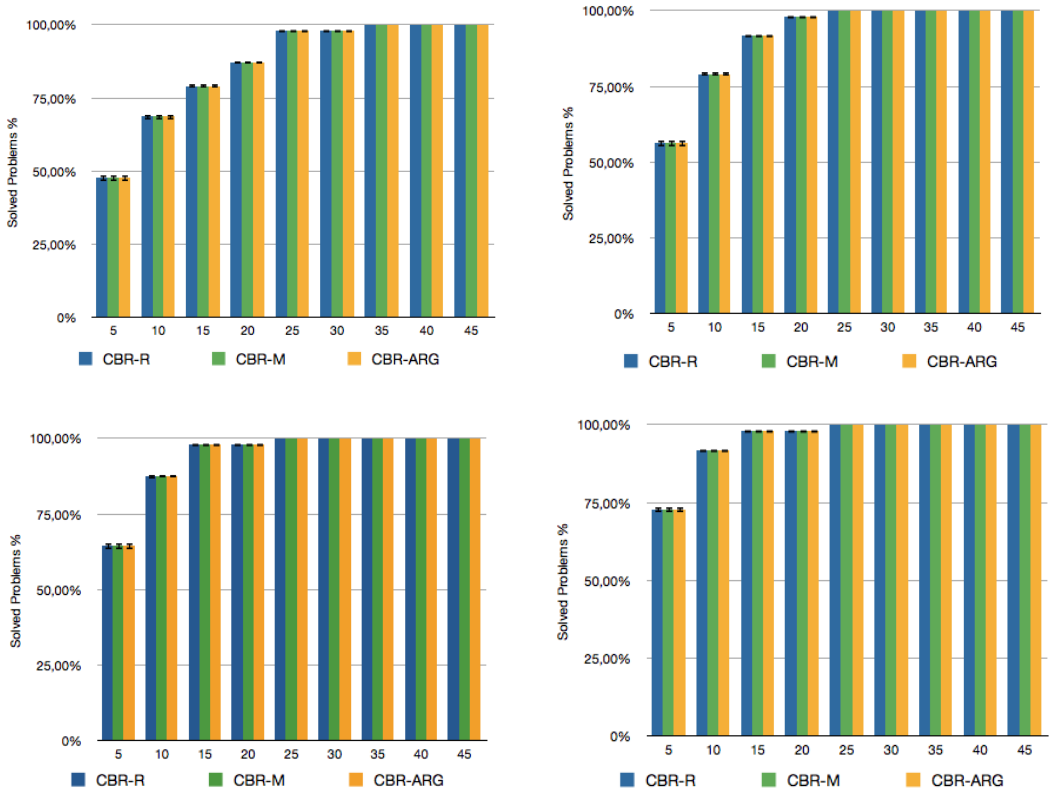

Figure 6.3: Percentage of Problems that are solved by 3 (to-left), 5 (top-right), 7 (bottom-left) and 9 (bottom-right) agents ([5, 45]Δ5 domain-cases; 20 argument-cases).

policy increases, we also achieve the same results for all policies, as shown in Figure 6.4. Therefore, this test demonstrates that the advantage of learning from argumentation experiences lies in the quality of the solution applied and not in the quantity of solutions achieved, as will be shown in the next section.

6.4.1.3 Percentage of problems that were properly solved with respect to the knowledge of the agents.

In this test, the percentage of problems that the system is able to solve, providing a correct solution, are computed. To check the solution accuracy, the solution agreed by the agents for each ticket requested is compared with its original solution, stored in the tickets case-base. One can expect that with more knowledge stored in the case-bases the number of problems that were correctly solved should increase. Figure 6.5

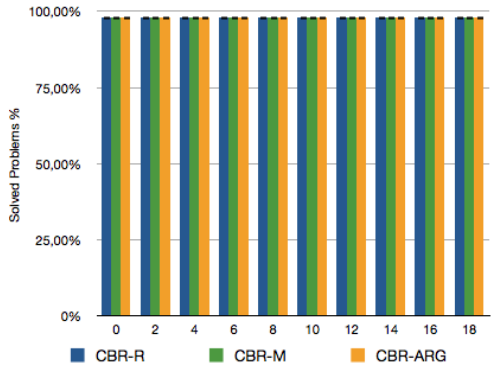

Figure 6.4: Percentage of Problems that are solved by 7 agents (20 domain-cases; [0, 18]Δ2 argument-cases).

shows how as the number of agents participating in the dialogue increases, the solution proposed by them is more appropriate and similar to the actual solution registered in the tickets case-base for the ticket that has been requested to the agents (the mean error percentage in the solution predicted decreases). Obviously, if more agents participate in the problem solving process, the probability that one or more of them have a suitable domain-case that can be used to provide a solution for the current problem increases. The same happens if the number of domain-cases of the agents case-base increases. This applies also in the case of the random policy, although this policy never achieves the 100% of correct solution predictions. Also, the results achieved by the argumentation policy improve those achieved by the other policies, even when the domain-cases case-bases are populated with a small number of cases. The argumentation policy achieves more than a 50% of improvement for a domain-cases case-base size up to 25 cases if 3 agents participate in the dialogue, up to 20 cases if 5 agents participate and up to 15 cases if there are 7 or 9 agents participating. These results demonstrate that if agents have the ability of arguing, the agents whose solutions are more supported by evidence have more possibilities of wining the argumentation dialogue and hence, the quality of the final solution selected among all potential solution proposed by the agents increases. Finally, Figure 6.6 shows the results of this test if the number of domain-cases is set to 20 and the number of argument-cases that the agents have is increased in each round. The results show that the argumentative knowledge has no substantial influence on the accuracy of the solution proposed, at least for the data used in this case of study.

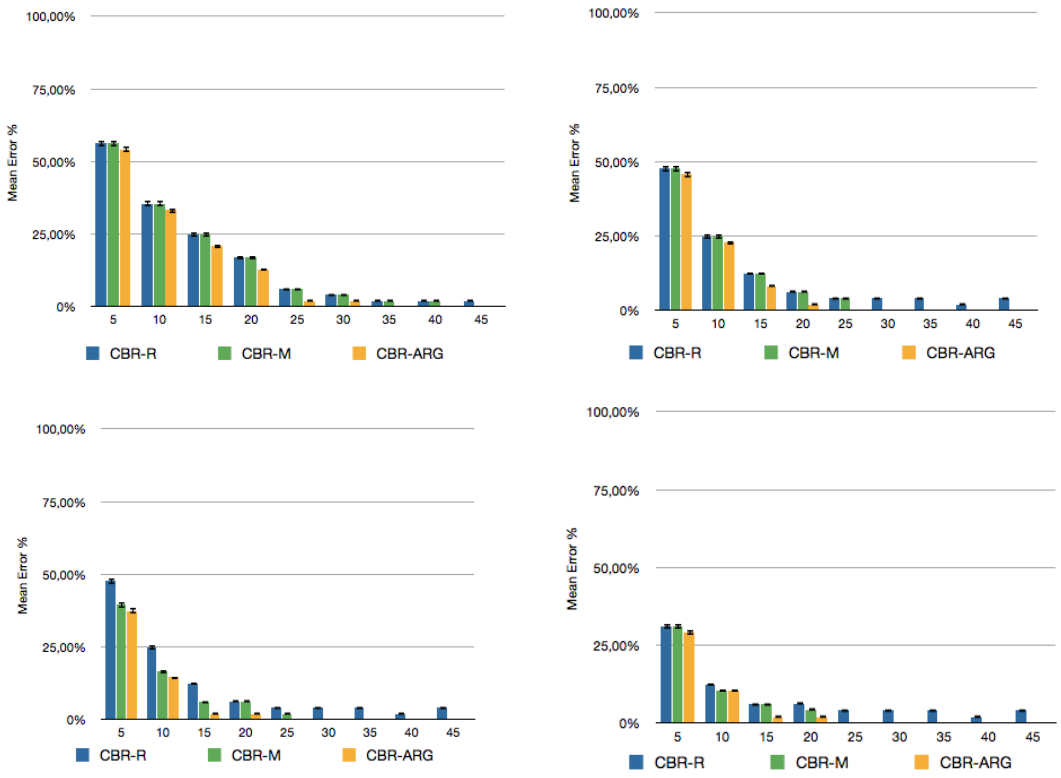

Figure 6.5: Solution prediction accuracy achieved by 3 (top-left), 5 (top-right), 7 (bottom-left) and 9 (bottom-right) agents ([5, 45]Δ5 domain-cases; 20 argument-cases).

6.4.1.4 Percentage of agreements reached with respect to the knowledge of the agents.

In this test, we evaluate the percentage of times that an agreement is reached and a frequency-based or a random choice among all possible solutions proposed by the agents is not necessary. Figure 6.7 shows the results obtained. For all policies, the overall trend of the agreement percentage is to increase as the knowledge about the domain that agents have increases. Nevertheless, figures show slight fluctuations between results. This behaviour can be explained since the addition of a small quantity of new domain-cases between two simulation rounds can give rise to temporary situations, such as some agents changing temporarily their opinions until new information is gained or obtaining the same *suitability degree* for several positions and arguments. In the last case random choices are made, which can have a slight negative effect on the overall performance of the system.

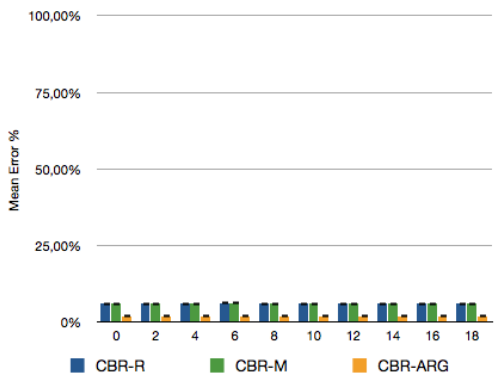

Figure 6.6: Solution prediction accuracy achieved by 7 agents (20 domain-cases; [0, 18]Δ2 argument-cases).

For the case of 3 agents, the small number of participants in the dialogue results in all policies achieving similar agreement percentages. However, when the number of agents grows up to 5, if agents follow an argumentative policy that allows them to argue and persuade other agents, those who have more support for their positions win the dialogue and convince the others to accept them. Thus, the percentage of agreements reached increases. In addition, the improvement on the agreement percentage grows more quickly for a larger number of agents, reaching more than the 80% with little knowledge about the domain (e.g. 10 domain-cases) for 7 and 9 agents participating in the dialogue. These results capture the fact that with more participating agents, the knowledge available among all of them to solve tickets increases and more useful argument-cases improve the performance of complex argumentation dialogues.

More interesting results can be observed if we compare the agreement percentage that the argumentation policy achieves when useful argument-cases are available. To perform this test, the percentage of agreement that agents reach in those cases that they have been able to find useful argument-cases (argument-cases which problem description matches the current situation) has been computed. Note that the fact that agents have argument-cases in their argument-cases case-bases does not necessarily mean that these cases match the current dialogue context and are actually used by the agents to make their decisions. Therefore, Figure 6.8 shows the percentage of agreements that the argumentation policy achieves when one or more agents use their argumentative knowledge. In these tests, the fluctuations between subsequent simulation rounds are notably greater than in the previous tests. These fluctuations are due to the fact that

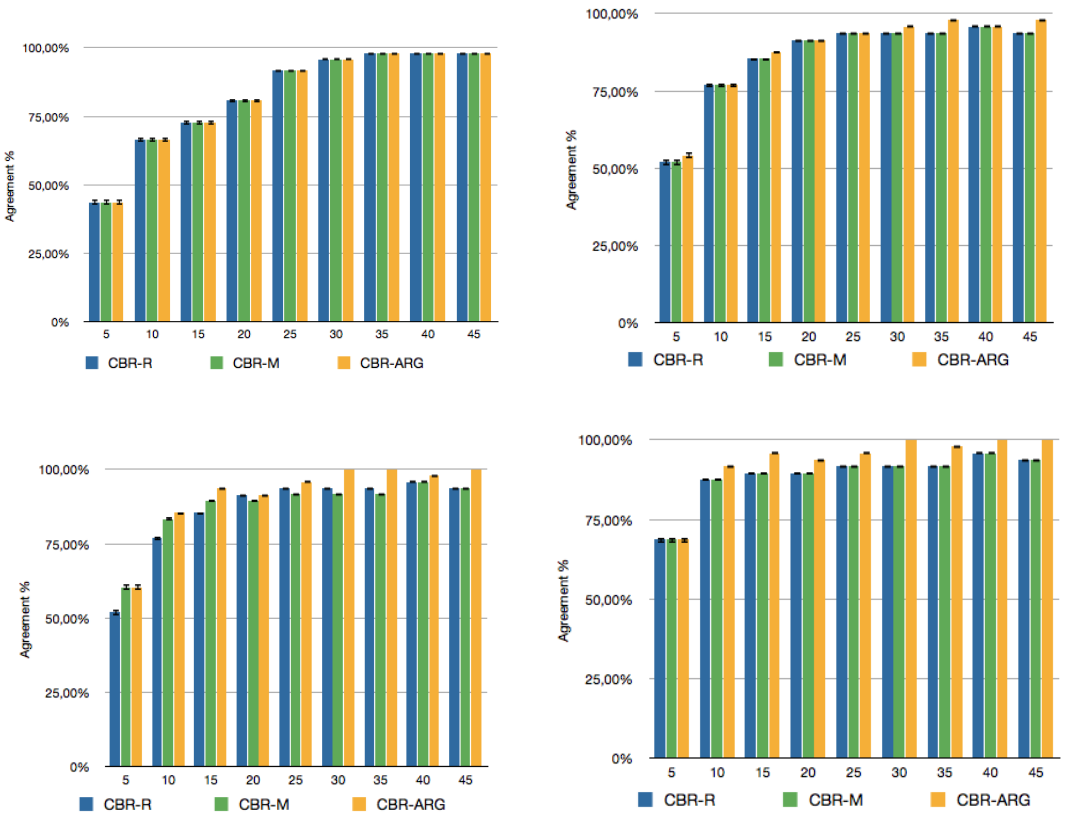

Figure 6.7: Percentage of agreement reached by 3 (top-left), 5 (top-right), 7 (bottom-left) and 9 (bottom-right) agents ([5, 45]Δ5 domain-cases; 20 argument-cases).

the percentage of useful argument-cases highly depends on the domain knowledge that agents have and on the dialogue context.

In the case of 3 agents, the small number of dialogue participants give rise to very simple dialogues and no argument-cases are actually used. This explains that the CBR-ARG policy gets the same results in the agreement percentage as the other policies, as shown in Figure 6.7. For 5, 7 and 9 agents, we can observe that when enough domain knowledge is available and agents engage in more complex dialogues (up to 30 domain-cases for 5 agents and up to 15 domain-cases for 7 and 9 agents), the agreement percentage has a global trend to increase when the initiator agent is the only agent that has useful argument-cases. This behaviour shows how the use of argumentative knowledge allows the initiator to argue better and persuade the other agents to accept their positions and reach an agreement. However, if more agents are also able to

improve their argumentation skills by using their argumentative knowledge (CBR-ARG AAC and CBR-ARG OAC policies), less agents are persuaded to accept other agents' positions and hence, no agreement is reached in almost all simulations (except for the case of 45 domain-cases in the agents domain-cases case-bases).

As in Figure 6.7, Figure 6.8 shows that when the number of agents that participate in the dialogue increases, the agreement percentage also increases for the CBR-ARG IAC policy. This can be observed by comparing the agreement percentage achieved between 5 and 7 agents. Between 7 and 9 agents, no significant changes in the agreement percentage for the CBR-ARG IAC policy are observed, while the CBR-ARG AAC and CBR-ARG OAC policies improve their results when agents have a high amount of domain knowledge. However, this increase has less to do with the use of argumentative knowledge than with the fact that more agents participate in the dialogue with almost full knowledge about the domain. Thus, most of them are able to provide the same accurate solution for the problem to solve.

Figure 6.8 also shows the average number of locutions interchanged among the agents during the argumentation dialogue. As expected, results demonstrate that more locutions are needed to solve tickets if there are more agents participating in the process. However, the number of interchanged locutions seems to stabilize when the percentage of agreements reached approaches to 100%. Also, when only one agent has argumentative knowledge, the number of locutions (or let us say, the number of dialogue steps) that are necessary to reach a final decision among agents is more stable than in the cases where more agents use their argument-cases. In fact, the dialogue steps in the cases of 7 and 9 agents are almost the same for this policy. Therefore, the CBR-ARG IAC policy is also the more efficient policy, achieving the best performance results with shorter argumentation dialogues among the agents.

Finally, to evaluate the influence of the amount of argumentative knowledge of the agents on the agreement percentage, Figure 6.9 shows the results obtained by the argumentation policy when the number of argument-cases available for one or more agents is increased. When the initiator agent is the only agent that uses argumentative knowledge, as this knowledge increases, the probability of finding useful argument-cases to apply in each argumentation dialogue also increases. Therefore, this agent improves its argumentation skills and it is able to persuade the others to reach an agreement and accept its position as the best solution to apply for the ticket to solve. However, when several agents have a small quantity of argument-cases, the probability of finding

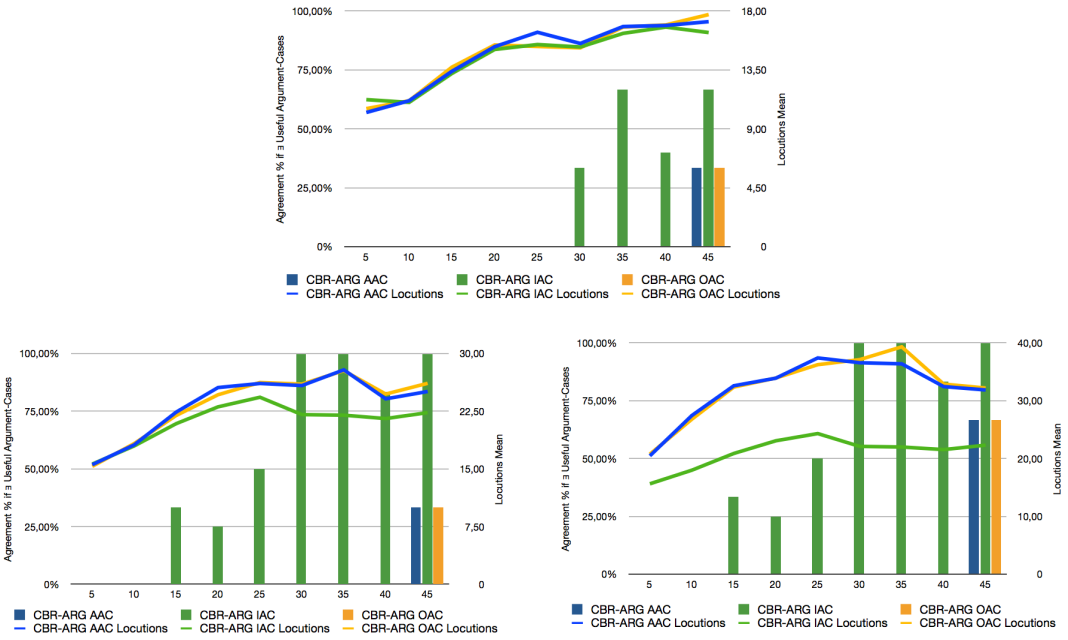

Figure 6.8: Percentage of agreement reached by 5 (top), 7 (bottom-left) and 9 (bottom-right) agents when useful argument-cases are available ([5, 45]Δ5 domain-cases; 20 argument-cases).

a useful argument-case is very low. In these cases (CBR-ARG AAC with 6 argument-cases and CBR-ARG OAC with 2 argument-cases), the performance of the system suffers from a high randomness, and this agent that finds a useful argument-case has a higher advantage over the others, being able to persuade them to reach an agreement that favours its preferences. Regarding the number of locutions interchanged among the agents, Figure 6.9 shows how the number of locutions to reach the agreement is stable for all policies and does not depend on the argumentation knowledge that agents have. Thus, as pointed out before, the CBR-ARG IAC policy gets higher percentage of agreement when useful argument-cases are actually used.

6.4.1.5 Percentage of agents that agree in the best solution to apply.

In this test, we evaluate the percentage of agents that agree in the best solution to apply when an agreement is reached. As shown in Figure 6.10, as the knowledge about the domain increases, all policies get higher percentages of agents that agree in the solution to apply. This expected behaviour is produced due to the fact that if agents have more knowledge to generate their positions, most accurate positions are generated and more

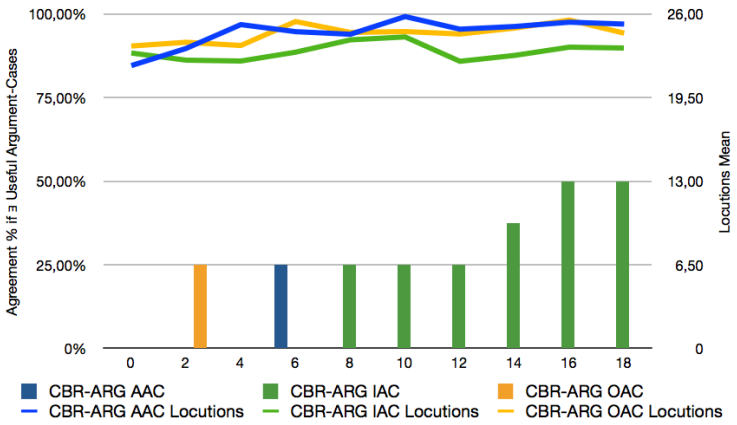

Figure 6.9: Percentage of agreement reached by 7 agents when useful argument-cases are available (20 domain-cases; [0, 18]Δ2 argument-cases).

agents coincide in their predictions. However, the CBR-ARG policy shows a more stable behaviour, showing an almost polynomial growth as the number of domain-cases that agents have increases.

Both CBR-R and CBR-M policies also improve their results as the number of agents participating in the dialogue increases, since, again, with more agents participating there are more information about the domain distributed across the agents' domain-cases case-bases and thus, there are more probability of agreeing in more accurate solutions. However, the CBR-ARG policy, despite getting higher percentages in the number of agents that agree than the other policies, presents a slightly variable behaviour, specially when agents have more than the 50% of domain knowledge (over 20 domain-cases). Thus, we can observe in Figure 6.10 that the argumentative policy gets more than the 50% of agents agreeing in the solution for the case of 3, 5 and 7 agents participating, while the percentage slightly decreases to exactly the 50% for the case of 9 agents. Similarly, for 35 domain cases the percentages fluctuate around the 75% of agents agreeing. This variable behaviour can arise from the influence that the number of useful argument-cases has in the results obtained fro the argumentative policy. Therefore, depending on the domain knowledege, which agents participate in the dialogue, and the contents of the argument-cases case-bases, more or less argument-cases can store useful information that matches the current dialogue. The amount of useful argument-cases has a direct influence on the persuasive power of the agents and thus, on the number of agents that agree in a specific solution. Nevertheless, the CBR-ARG

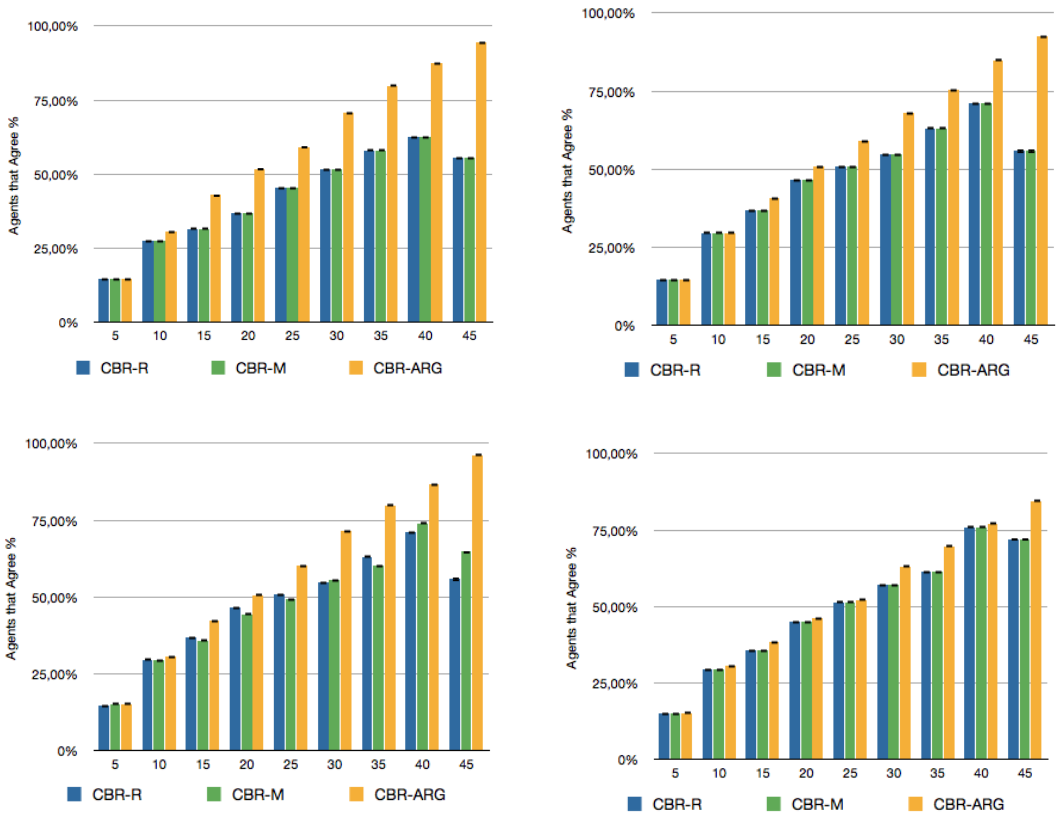

Figure 6.10: Percentage of agents that agree among 3 (top-left), 5 (top-right), 7 (bottom-left) and 9 (bottom-right) agents ([5, 45]Δ5 domain-cases; 20 argument-cases).

policy still obtains the same or better results than the CBR-R and the CBR-M policies in all cases.

To evaluate the influence that useful argumentative knowledge has in the percentage of agents that agree in a solution, Figure 6.11 shows the results of this test when agents are able to find argument-cases in their argument-cases case-bases that match the current dialogue context. In the case of 3 agents participating in the dialogue, no matching argument-cases are found, due to the little number of participants resulting in simple argumentation dialogues. However, as the number of participants and domain knowledge increases, if one agent has useful argumentative knowledge, it effectively persuades the other agents to accept its position as the best solution to apply to solve the ticket requested and the results of policy CBR-ARG IAC show a clear growing trend. However, if all or most agents agents have useful argument-cases in their argument-

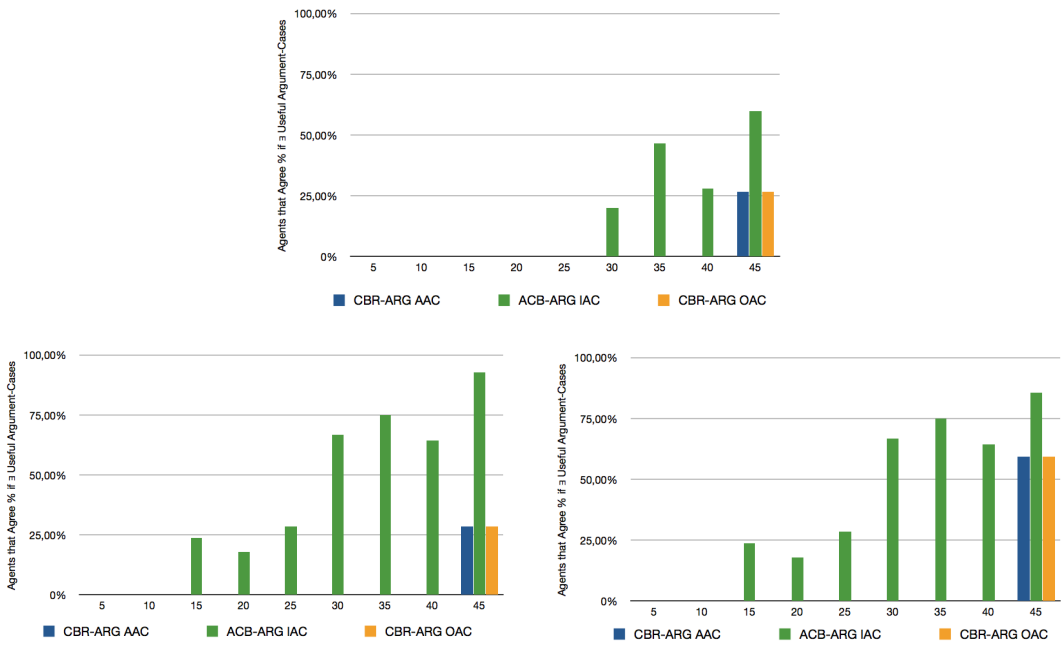

Figure 6.11: Percentage of agents that agree among 5 (top), 7 (bottom-left) and 9 (bottom-right) agents when useful argument-cases are available ([5, 45]Δ5 domain-cases; 20 argument-cases).

cases case-bases, their persuasive power is the same and they do not agree in a unique solution to apply until they have almost full knowledge about the domain (45 domain-cases). This can be observed in the results shown in Figure 6.11 for the CBR-ARG AAC and the CBR-ARG OAC policies.

Finally, to evaluate the influence of the amount of argumentative knowledge of the agents on the percentage of agents that agree in the final solution, Figure 6.12 shows the results obtained by the argumentation policy when the number of argument-cases available for one or more agents is increased. When the initiator agent is the only agent that uses argumentative knowledge, as this knowledge increases, the probability of finding useful argument-cases to apply in each argumentation dialogue also increases. Therefore, this agent improves its argumentation skills and its persuasive power to convince the other agents to agree in a unique solution (its position) as the best to solve the ticket requested. However, when several agents have a small quantity of argument-cases, the probability of finding a useful argument-case is very low. In these cases, as for the results shown in Figure 6.9 for the agreement percentage, the performance of the system suffers from a high randomness, and this agent that finds a useful argument-

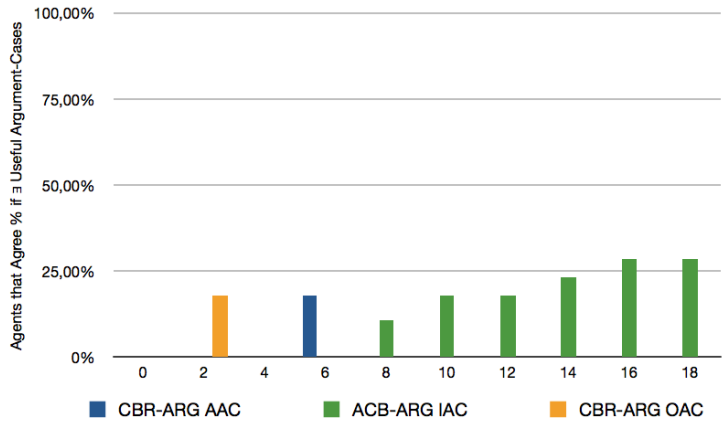

Figure 6.12: Percentage of agents that agree among 7 agents when useful argument-cases are available (20 domain-cases; [0, 18]Δ2 argument-cases).

case has a bigger advantage over the others, being able to persuade them to reach an agreement that favours its preferences. This explains the results obtained by the CBR-ARG AAC policy with 6 argument-cases and the CBR-ARG OAC policy with 2 argument-cases.

6.4.1.6 Percentage of positions accepted with respect to the number of argument-cases.

This test evaluates the percentage of position that an agent (the initiator, for instance) gets accepted by the other agents in different settings of the argumentative policy, when useful argument-cases are used. The CBR-R and the CBR-M policies do not allow agents to argue and hence, they do not accept or defeat the position of other agents. Therefore, these policies are not considered for this test. Figure 6.13 shows how, independently of the number of agents participating in the dialogue, once the knowledge about the domain overpasses certain threshold (15 domain-cases), if an agent has argument-cases that match the current dialogue context, it gets its position accepted, even if the other participants have also useful argument-cases. However, if this agent does not have argumentative knowledge, but the other agents does, the percentage of acceptance depends on how useful the argumentative knowledge of the others is and it varies until the agent has enough domain knowledge to propose a good enough solution, as shown by the results obtained by the CBR-ARG OAC policy.

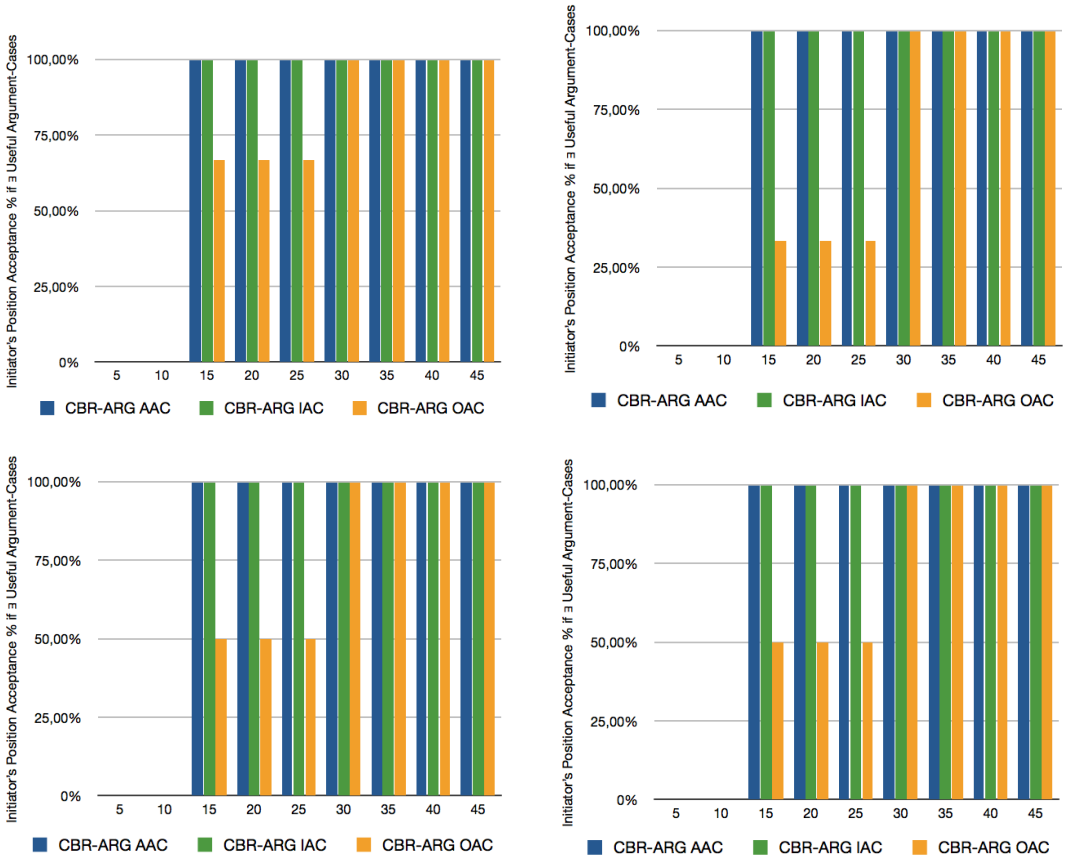

Figure 6.13: Percentage of positions accepted for 3 (top-left), 5 (top-right), 7 (bottom-left) and 9 (bottom-right) agents when useful argument-cases are available ([5, 45]Δ5 domain-cases; 20 argument-cases).

Finally, to evaluate the influence of the amount of argumentative knowledge that an agent has in the percentage of acceptance of its positions, Figure 6.14 shows the results obtained by setting the number of domain-cases to 20 and increasing the number of argument-cases by 2 in each round. Results show that if only one agent has argumentative knowledge (CBR-ARG IAC policy), once it has argument-cases that apply to the current dialogue situation and enough domain knowledge, it gets its position accepted. If other agents also have argumentative knowledge (CBR-ARG AAC policy), the percentage of acceptance varies until enough useful argument-cases can be used (from 10 forwards), since this percentage depends also on how good are the argumentation skills of the other participants. Finally, if all participants have useful argumentative knowledge (CBR-ARG OAC policy), the percentage of acceptance of the initiator's position

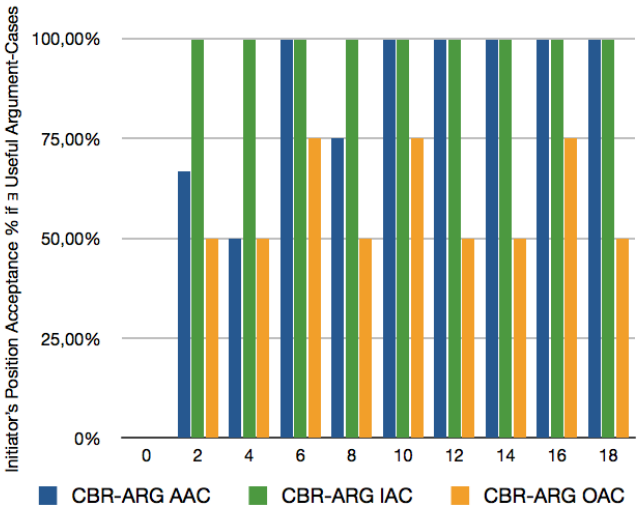

Figure 6.14: Percentage of positions accepted for 7 agents (20 domain-cases; [0, 18]Δ2 argument-cases).

depends on which agent is able to create more persuasive arguments and convince the others to accept its position as the best to solve the requested ticket.

6.4.2 Testing the Argumentation Strategies

In the following tests, we have compared different argumentation strategies (as combinations of an agent profile and different tactics) that an agent can follow to argue with other agents. The argumentative profile presented in Chapter 4 complies with the reasoning process and dialogue protocol proposed for the case-based argumentation framework presented in this thesis. Other agent profiles involve different modifications of them. Therefore, to perform these tests, an agent (say the initiator agent) has been set to have an *argumentative* profile and to follow different tactics to argue with agents of different profiles. Also, these agents do not follow any specific tactic. As in the performance tests, by default, all agents know each other, all are in the same group and the dependency relation between them is *charity*. The values of each agent have been randomly assigned and agents know the values of their partners. Also, all agents play the role of *operator*. In addition, agents assign weights to the *similarity degree* (w_{SimD}) and the *support factor* (w_{SF}) proportionally to the number of domain-cases and argument-cases that they have in their case-bases.

The different strategies evaluated represent the combination of the argumentative profile for the initiator with the different tactics proposed in Chapter 4:

ST1 Persuasive Strategy: with this strategy, the agent selects such positions and arguments whose associated argument-cases were more persuasive in the past (have more persuasiveness degree).

ST2 Maximise-Support Strategy: with this strategy, the agent selects such positions and arguments that have higher probability of being accepted at the end of the dialogue (their associated argument-cases have more support degree).

ST3 Minimise-Risk Strategy: with this strategy, the agent selects such position and arguments that have a lower probability of being attacked (their associated argument-cases have less risk degree).

ST4 Minimise-Attack Strategy: with this strategy, the agent selects such positions and arguments that have received a lower number of attacks in the past (their associated argument-cases have less attack degree).

ST5 Maximise-Efficiency Strategy: with this strategy, the agent selects such positions and arguments that lead to shorter argumentation dialogues (their associated argument-cases have higher efficiency degree).

ST6 Explanatory Strategy: with this strategy, the agent selects such positions and arguments that cover a bigger number of domain-cases, argument-cases or dialogue graphs. That is, the positions that are similar to argument-cases that have more justification elements.

These strategies are evaluated by computing the agreement percentage obtained by the agents when they have a *low* (from 5 to 15 domain-cases), *medium* (from 20 to 30 domain-cases) or *high* (from 35 to 45 domain-cases) knowledge about the domain. Also, the argumentative agent has a full case-base of 20 argument-cases. Then, the percentage of agents that an argumentative agent (the initiator, for instance) is able to persuade to reach an agreement to propose its solution as the best option to solve a ticket is computed for the same settings (from low to high knowledge about the domain). To be able to follow effectively an argumentation tactic, an agent needs to have useful argument-cases to reuse these experiences for the current argumentation situation. Thus, all strategic tests report results obtained when the initiator agent has

some argument-cases that match the current situation and actually uses them to select the best position and arguments to propose. The following tables show the results obtained.

Table 6.3 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have an *agreeable* profile. Also, Table 6.4 shows the percentage of agreeable agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. Agreeable agents do not challenge the positions of other agents, but only try to defend their positions if they are attacked. However, an agreeable agent accepts the position of other agent that has proposed its same position or a position that it has generated, although it has not ranked it as the best solution to apply. That means that in these cases, the agreeable agent votes the position of other agents and withdraws its own.

For low knowledge about the domain, no useful argument-cases are found and agents cannot reach an agreement about the best solution to apply to solve the ticket at hand, whatever strategy is followed. This does not means that the system is not able to propose a solution at all, but the solution proposed is not agreed by all agents. In the case of having a medium amount of knowledge about the domain, the best results are achieved for these strategies that minimise the probability or the number of potential attacks that an argument can receive (ST3 and ST4). Remember that an agreeable agent does not attack any position and if attacked, if it cannot defend itself, it just withdraw its position. Therefore, if the initiator uses arguments to generate its position and arguments that the agreeable agent cannot defeat, the initiator will have less agreeable agents competing in the dialogues and an agreement is reached more easily. However, if the agents have more knowledge about the domain, the agreeable agents increase their options to generate potential attacks to rebut the attacks that they receive, and hence, following a strategy that selects those positions and arguments that are expected to have a higher acceptability degree from the other agents (ST1) makes the initiator to be able to reach most agreements.

The results of Table 6.4 show again how ST3 and ST4 perform better for a medium amount of knowledge about the domain and ST1 for a high amount of knowledge. This reinforces our hypothesis that these strategies make the initiator agent to reach to a higher number of agreements by persuading other agents to accept its position as the best solution to apply for a ticket requested.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	25.00%	16.67%	*33.33%*	*33.33%*	0.00%	16.67%
HIGH	*63.89%*	55.56%	50.00%	25.00%	38.89%	19.44%

Table 6.3: Agreement Percentage with Agreeable agents.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	14.29%	11.90%	*22.62%*	*22.62%*	0.00%	11.90%
HIGH	*52.38%*	45.24%	38.89%	21.03%	30.95%	14.68%

Table 6.4: Percentage of Agreeable agents persuaded.

Table 6.5 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have a *disagreeable* profile. In addition, Table 6.6 shows the percentage of agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket when it is arguing with disagreeable agents. Disagreeable agents act similarly to agreeable agents, do not challenging any position proposed by other agents, but in this case, this profile of agents only accept the position of a partner (voting it and withdrawing its own), if it exactly coincides with the position that they have proposed as the best solution to apply.

Again, low knowledge about the domain prevents the use of useful argument-cases. However, for both medium and high amounts of knowledge about the domain, the initiator agent is able to convince more agents if it follows an strategy that minimises the number of potential attacks that its position and arguments can receive, getting thus to higher agreement percentages. This results in selecting those positions that, although still serve to the initiator's agent objectives, are more similar to the positions proposed by the disagreeable agents and hence, these have less potential attacks to put forward and are more easily persuaded to reach an agreement to accept the initiator's position (as shown in Table 6.6). Nevertheless, disagreeable agents are difficult to convince and both agreements and agents that agree percentages are low independently of the strategy followed by the initiator.

Table 6.7 shows the percentage of times that an agreement about the best solution

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	16.67%	0.00%	8.33%	*25.00%*	8.33%	16.67%
HIGH	36.11%	30.56%	11.11%	*38.89%*	30.56%	25.00%

Table 6.5: Agreement Percentage with Disagreeable agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	11.90%	0.00%	5.95%	*15.48%*	5.95%	11.90%
HIGH	30.56%	25.40%	9.52%	*31.35%*	25.79%	23.02%

Table 6.6: Percentage of Disagreeable agents persuaded.

to apply is reached when the argumentative agent is arguing with agents that have an *elephant's child* profile. In addition, Table 6.8 shows the percentage of elephant's child agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. Elephant's child agents have a weird profile that challenges any position proposed by other agents, whatever this position is. This agents are used to overload the system with useless interactions between the agents. An elephant's child agent only accepts the position of other agent if it challenges it and the other agent wins the debate. This has more probability of occurring when agents have less knowledge about the domain and thus, less knowledge to generate positions and arguments. Therefore, whatever strategy the initiator follows, the percentage of agreement overpass only the 50% if agents have low knowledge about the domain and the initiator is able to persuade the 28.57% of agents, as shown in the tables, and decreases as the amount of domain knowledge increases. However, in most cases elephant's child agents are attacking positions that they are able to generate and support and thus, the attacked agents are defeated and withdraw their positions, which prevents the effective development of agreement processes, specially when agents have more domain knowledge and are able to propose more solutions.

Table 6.9 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have an *open-minded* profile. Also, Table 6.10 shows the percentage of open-minded agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. An open-minded agent only challenges the positions

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6
LOW	66.67%	66.67%	66.67%	66.67%	66.67%	66.67%
MEDIUM	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%
HIGH	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 6.7: Agreement Percentage with Elephant's Child agents.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6
LOW	28.57%	28.57%	28.57%	28.57%	28.57%	28.57%
MEDIUM	15.48%	15.48%	15.48%	15.48%	15.48%	15.48%
HIGH	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 6.8: Percentage of Elephant's Child agents persuaded.

of other agents if it has not been able to generate such positions (they are not its proposed position or a position in its list of potential positions to propose). However, what specially distinguishes the behaviour of this agent from other agent profiles is the fact that it accepts the position of an agent that has started and won a dialogue with it, although this position is not in its list of potential positions.

As for agreeable and disagreeable agents, if the knowledge that the agents have about the domain is low, no agreement is reached. For the case of having a medium amount of domain knowledge, the agreement is reached easily if the initiator follows a strategy that selects the most potentially persuasive arguments or those that cover as much justification elements as possible (ST1 and ST6). If the initiator is able to better defend its position with more persuasive arguments or more support elements, it ensures that its position will prevail accepted and thus, the defeated open-minded agents will withdraw their positions and agree to propose the initiator's as the best solution to apply. Thus, Table 6.10 shows a higher percentage of agents persuaded if the initiator follows ST1 and ST6. However, if agents have high knowledge about the domain, the percentage of agreements and agents persuaded are clearly higher when the initiator follows ST2 and ST6. This demonstrates that when the initiator agent has more knowledge and hence, more support degree (higher probability of being accepted at the end of the dialogue) and more elements to justify its positions and arguments, open-minded agents easily accept the position of the initiator and withdraw theirs when they attack the initiator's position.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	*41.67%*	33.33%	25.00%	33.33%	16.67%	*41.67%*
HIGH	47.78%	61.11%	47.78%	52.22%	50.00%	*62.78%*

Table 6.9: Agreement Percentage with Open-Minded agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	*28.57%*	23.81%	16.67%	23.81%	11.90%	27.38%
HIGH	37.46%	48.57%	37.86%	44.21%	38.89%	*51.19%*

Table 6.10: Percentage of Open-Minded agents persuaded.

Table 6.11 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that also have an *argumentative* profile. Also, Table 6.12 shows the percentage of agents with its same profile that an argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. An argumentative agent only challenges positions of other agents when they have proposed a position different from its position. In addition, this agent profile accepts such positions that it attacks when the opponent wins the confrontation.

Again, as in most strategy tests, low knowledge about the domain results in simple dialogues that do not reach any agreement and the solution proposed is not agreed by all agents. For a medium amount of knowledge about the domain, the percentage of agreements reached is higher when the initiator follows a strategy that minimises the probability that their positions and arguments are attacked (ST3). In fact, if positions are not attacked at all, they prevail as potential candidates to be selected as the final solution for the ticket requested. Note that in this test, all agents accept positions and arguments of other agents under the same circumstances, so decreasing the probability of a position to be attacked, increases its probability of being accepted at the end of the dialogue.

If argumentative agents have a high knowledge about the domain, many of them are able to propose accurate solutions and defend them against attacks. Therefore, such strategies that allow the initiator to better defend its positions and arguments by in-

creasing the probability of being accepted (ST2) or by reducing the number of potential attacks get better agreement percentages (ST4). If we understand that more acceptable and less attackable arguments lead to shorter dialogues, the good results achieved when the initiator follows a strategy that selects those positions and arguments that produced shorter dialogues in the past (ST5) can be viewed as a logical consequence of the good performance of ST2 and ST4. Table 6.12 shows a slightly higher percentage of persuaded agents when the initiator has deep knowledge about the domain and the number of potential attacks is decreased. As less attacks are received, less effort the initiator needs to convince other agents that attack its position to agree and accept it as the best solution to apply for the requested ticket. Nevertheless, all ST2, ST4 and ST5 strategies get good percentages of persuaded agents for this amount of domain knowledge.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	38.89%	47.22%	*50.00%*	33.33%	33.33%	41.67%
HIGH	82.22%	*100.00%*	82.22%	*100.00%*	*100.00%*	86.11%

Table 6.11: Agreement Percentage with Argumentative agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	22.22%	29.76%	*30.95%*	21.43%	23.81%	29.76%
HIGH	68.57%	80.56%	66.98%	*81.35%*	80.56%	71.43%

Table 6.12: Percentage of Argumentative agents persuaded.

Summarizing, Tables 6.13 and 6.14 show, for each strategy, the agent profiles that have achieved in average (for all amounts of domain knowledge) the highest percentage of agreement and the highest percentage of agents that the argumentative agent is able to persuade, respectively. For all strategies, the best results in the number of agreements reached and agents persuaded are obtained when all agents are argumentative. Among them, the strategy that guides the agent to propose positions and arguments that have a higher potential probability of being accepted (ST2) is the winning strategy for our customer support domain, followed closely by the strategy that makes the agent to minimise the number of potential attacks that its positions and arguments can receive (ST4). It seems reasonable that if we are measuring the percentage of agreements

reached and the percentage of agents persuaded to reach them, that strategy that increases the acceptance probability of positions and arguments performs better. Also, if agents receive less number of attacks, it can be understood as a consequence of proposing positions and arguments that suit the objectives of more agents.

For the rest of profiles, agreeable agents are more easily persuaded with positions and arguments that have a low probability of being attacked (probably because agreeable agents are also able to generate them); disagreeable agents are more easily persuaded with positions and arguments that will potentially get less number of attacks (probably because they are the same that disagreeable agents have proposed); and open-minded agents are easily persuaded with positions and arguments that have more elements that justify them (and hence the initiator has more elements to rebut attacks and win the dialogue). As pointed out before, elephant's child agents show a weird behaviour that gets low percentages of agreement and agents persuaded no matter which strategy the initiator follows. However, in most cases elephant's child agents are attacking positions that they are able to generate and support and thus, the attacked agents are defeated and withdraw their positions, which prevents the development of an effective agreement process.

Strategy / Agent Profile	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
Agreeable	38.10%	30.95%	*35.71%*	25.00%	16.67%	19.05%
Disagreeable	29.76%	13.10%	8.33%	*34.52%*	16.67%	17.86%
Elephant's Child	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%
Open-Minded	41.90%	47.62%	38.33%	40.24%	35.71%	*48.33%*
Argumentative	**51.90%**	**66.67%**	**60.24%**	**64.29%**	**57.14%**	**58.33%**

Table 6.13: Average agreement percentage for all agent profiles.

Strategy / Agent Profile	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**
Agreeable	28.57%	24.49%	*26.36%*	18.71%	13.27%	12.93%
Disagreeable	22.28%	10.88%	6.63%	*24.15%*	13.61%	14.97%
Elephant's Child	6.63%	6.63%	6.63%	6.63%	6.63%	6.63%
Open-Minded	30.85%	35.10%	27.45%	31.70%	25.85%	*36.22%*
Argumentative	**38.91%**	**49.83%**	**44.52%**	**48.13%**	**44.73%**	**45.92%**

Table 6.14: Average percentage of agents persuaded for all agent profiles.

6.4.3 Testing the Social Context

The ability of the framework to represent the social context of the system has also been evaluated. To perform these tests, the system has been executed with 7 participating agents, following the argumentative policy that complies with the proposals of this thesis (CBR-ARG). These settings are selected by taking into account the results of the performance tests, which show that this configuration allows agents to argue with fair enough information to provide suitable positions and arguments, but leaving room for the argumentation to make sense. The knowledge about the domain that each agent has is increased by 5 domain-cases in each round, from 5 to 45 domain-cases. Argumentative agents have a full argument-cases case-base populated with 20 cases. By default, all agents know each other, all are in the same group and the dependency relation between them depends on the specific test. The influence of different degrees of friendship and group membership are difficult to evaluate with the limited amount of data of our tickets case-base and remains future work. The values of each agent have been randomly assigned from a set of pre-defined values (*efficiency* of the problem solving process, *accuracy* of the solution provided and *savings* in the resources used to solve the ticket).

In addition, argumentative agents assign weights to the *similarity degree* (w_{SimD}) and the *support factor* (w_{SF}) proportionally to the relation between the number of domain-cases and argument-cases that they have in their case-bases, as it was done in the performance tests. Also, by default agents do not follow any dialogue strategy, setting the same weight for all elements of the support factor.

Following, the influence of the presence of an expert, the presence of a manager, and the knowledge about the values of other agents in the system performance is evaluated.

6.4.3.1 Presence of an Expert

In this test, an agent has been allowed to play the role of an *expert*, while the rest of agents play the role of *operators*. An expert is an agent that has specific knowledge to solve certain types (categories) of problems and has its case-base of domain-cases populated with cases that solve them. Thus, the expert domain-cases case-base has as much knowledge as possible about the solution of past problems of the same type. That is, if the expert is configured to have 5 domain-cases in its domain-cases case-base, and there are enough suitable information in the original tickets case-base, these

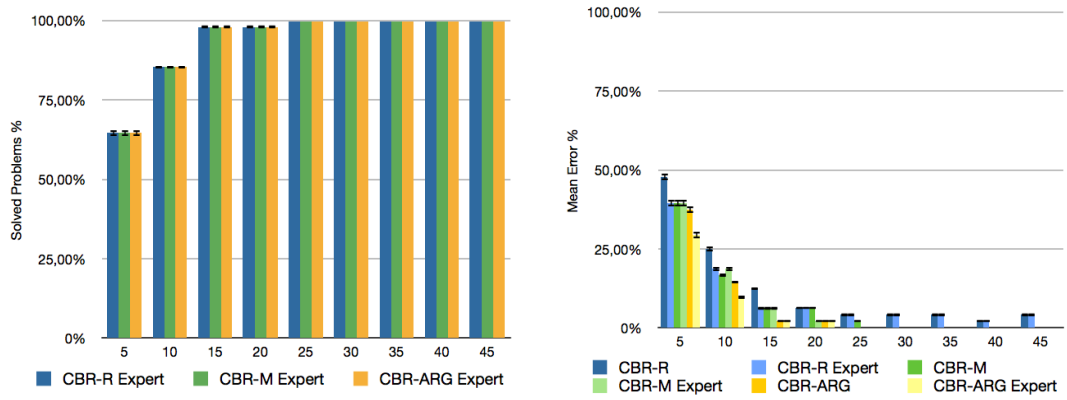

Figure 6.15: Percentage of problems that are solved by 1 expert and 6 operators (left) and accuracy of their predictions (right) ([5, 45]Δ5 domain-cases; 20 argument-cases).

cases represent instances of the same type of problems. In the case that the tickets case-base has less than 5 cases representing such category of problems, 3 for instance, the remaining two cases are of the same category (if possible).

In our case, the expert agent has an *authorisation* dependency relation over other technicians. Therefore, if it is able to propose a solution for the ticket requested, it can generate arguments that support its position and that will defeat other operators' arguments, due to the *defeat relation* among arguments defined in Chapter 3. This relation assigns more importance to the arguments of an agent that has an authorisation dependency relation over other agents.

All simulation tests have been executed and their results compared for the random based decision policy (CBR-R Expert), the majority based decision policy (CBR-M Expert) and the argumentation based policy (CBR-ARG Expert). For these policies, the domain-cases case-base of the expert has been populated with expert domain knowledge. To evaluate the global effect of this expert knowledge, the results obtained for the accuracy of predictions when the domain-cases case-base of all agents are populated with random data are also shown for each policy (CBR-R, CBR-M and CBR-ARG).

Figure 6.15 shows how all policies are able to solve the same percentage of problems, but the accuracy of predictions is higher if agents are allowed to argue following the CBR-ARG Expert policy. Comparing the results obtained when the initiator has (CBR-R Expert, CBR-M Expert and CBR-ARG Expert) or does not have expert knowledge (CBR-R, CBR-M and CBR-ARG), as expected, agents are able to reach better ac-

curacy in their final prediction when they are able to argue and there is an expert participating in the argumentation dialogue (CBR-ARG Expert). This demonstrates that the decisions of the expert prevail and, as it has more specialised domain-knowledge to propose solutions, the predictions of the system are more accurate.

6.4.3.2 Presence of a Manager

In this test, an agent has been allowed to play the role of a *manager*, other agent plays the role of *expert* (with the same configuration for its domain-cases case-base as in the previous test) and the other agents play the role of *operators*. A manager is an agent which decisions have to be observed by other agents with an inferior role in the system, due to its higher position in the company hierarchy of roles. A manager can be an expert, with specialised knowledge about the best solution to apply to specific types of problems, or else, a technician that has access to more information than an operator, due to its highest rank in the company. To simulate the presence of a manager in the system, we have considered the last case and populated the manager domain-cases case-base with twice the information that the domain-cases case-base of the operators (until a maximum amount of 45 domain-case).

In our case, the manager agent has a *power* dependency relation over other technicians. Therefore, if it is able to propose a solution for the ticket requested, it can generate arguments that support its position and that will defeat other technicians' arguments, due to the *defeat relation* among arguments defined in Chapter 3. This relation assigns more importance to the arguments of an agent that has a power dependency relation over other agents.

All simulation tests have been executed and their results compared for the random based decision policy (CBR-R Manager), the majority based decision policy (CBR-M Manager) and the argumentation based policy (CBR-ARG Manager). For these policies, the domain-cases case-base of the manager has been populated with more domain knowledge and the domain-cases case-base of the expert agent (other agent selected randomly) bas been populated with expert knowledge, as explained before. To evaluate the global effect of these different configurations of the domain knowledge, the results obtained for the accuracy of predictions when the domain-cases case-base of all agents are populated with random data are also shown for each policy (CBR-R, CBR-M and CBR-ARG).

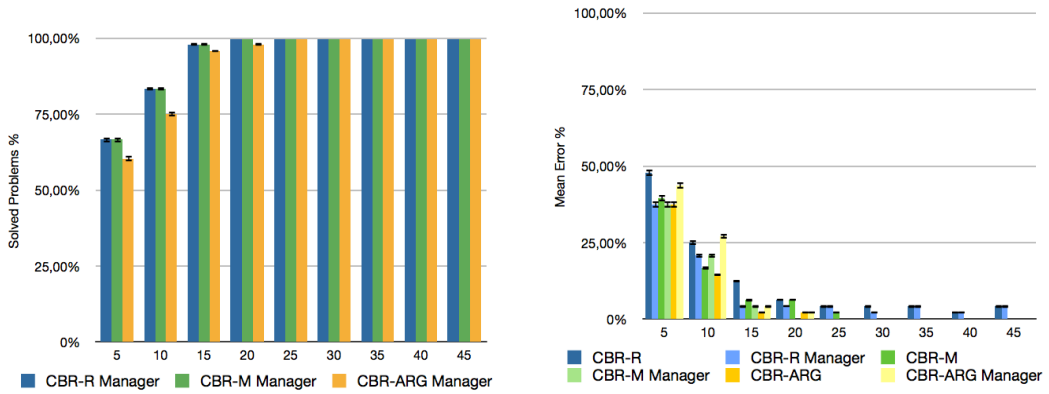

Figure 6.16: Percentage of problems that are solved by 1 manager and 6 operators (left) and accuracy of their predictions (right) ([5, 45]Δ5 domain-cases; 20 argument-cases).

Figure 6.16 shows that although the manager has more amount of domain knowledge, as this in not specialised knowledge to solve any type of problems, its predictions can have less quality than the operators' predictions. However, as the defeat relation of our case-based argumentation framework assigns higher priority to its decisions (even when there is an expert participating in the dialogue), these low quality solutions are accepted as the final solutions that the system proposes. Therefore, until all agents have enough amount of domain knowledge to provide accurate solutions (up to 20 domain-cases), the argumentative policy gets worse results both in percentage of problems solved and in mean error than the other policies. In fact, the percentage of problems that the system is able to solve is lower when agents argue and have low knowledge about the domain. Also, when agents have low knowledge about the domain, as the positions of the manager are selected over the expert's positions, the systems gets a poor accuracy in its predictions, quite worse than in the case that all agents have the same amount of random knowledge about the domain.

6.4.3.3 Knowledge about Other Agents' Social Context

With these tests, we have evaluated the influence that the knowledge about the social context has in the performance of the system. Therefore, we have compared the performance of the system when the participating agents follow an argumentation policy and have full information about the social context of their partners (CBR-ARG), or on the contrary, do not know the preference over values that their partners have

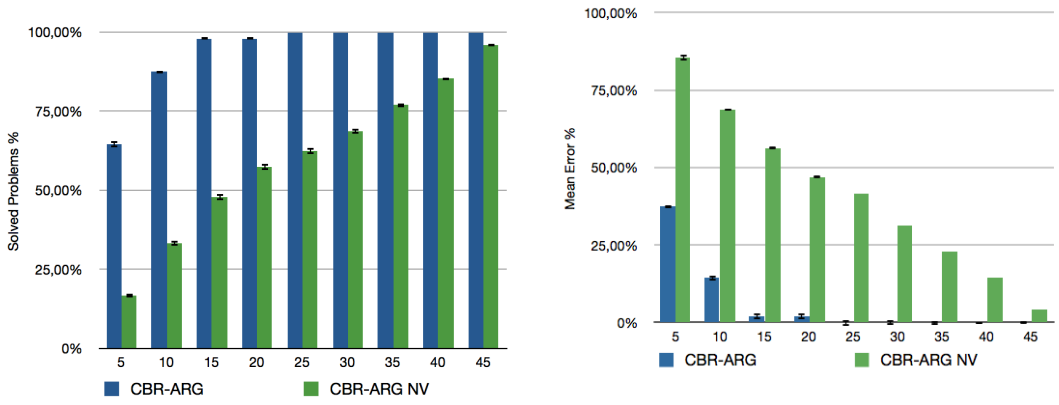

Figure 6.17: Percentage of problems that are solved by 7 agents (left) and accuracy of their predictions (right) ([5, 45]Δ5 domain-cases; 20 argument-cases).

(CBR-ARG NV). In a real company, the dependency relations over technicians and the group that they belong are known by the staff. Hence, we assume that agents know this information about their partners.

In our evaluation domain, an agent assigns the same importance (weight) to both domain and argumentation knowledge to generate and select positions and arguments. However, if it does not know the value preferences of their partners, many times the agent uses argument-cases that are not suitable for the current situation. This makes the agent to make wrong decisions that worsen the global performance of the system.

Figure 6.17 shows clearly that the performance of the system is negatively affected when argumentative agents use incorrect argument-cases to make their decisions. Then, for instance, the percentage of solved problems that argumentative agent are able to solve when they have full knowledge about the social context of their partners reaches almost the 100% with low knowledge about the domain (15 domain-cases), while it barely reaches the 50% when agents do not know the values of their partners. Similarly, the prediction error for well-informed agents is almost null with 15 domain-cases, while it still has a 5% error percentage with a high amount of knowledge about the domain (45 domain-cases) when agents ignore the values of the other agents.

Finally, Figure 6.18 shows also how system presents a poor performance in terms of the agreement percentage and the percentage of agents that agree when argumentative agents ignore the values of their partners. Again, the use of wrong argument-cases makes argumentative agents to propose solutions and arguments that hinder to reach

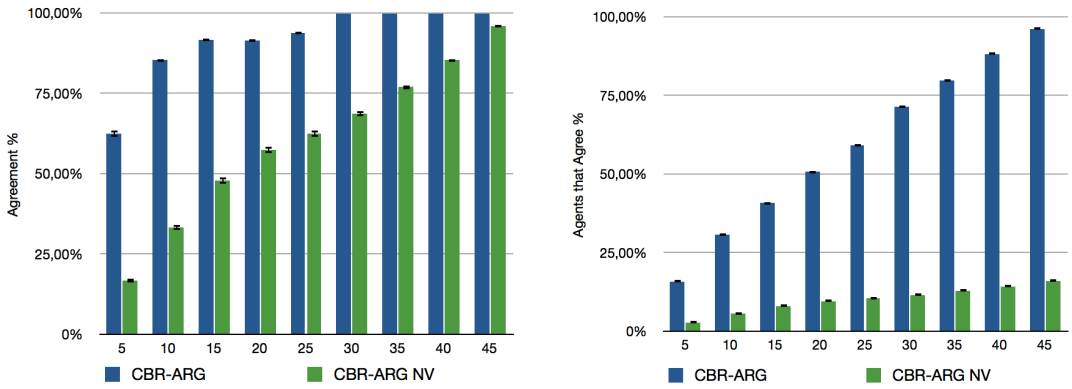

Figure 6.18: Percentage of agreement reached by 7 agents (left) and percentage of agents that agree (right) ([5, 45]Δ5 domain-cases; 20 argument-cases).

agreements. This could be avoided if the system assigns less importance to the argumentative knowledge, by reducing the weight of the support factor (w_{SF}). In this way, a system that supports our framework can also perform well in domains where acquiring social context information about competitors is difficult, although this would significantly reduce the advantages of learning this type of information.

6.5 Conclusions

In this chapter, several tests to evaluate the case-based argumentation framework proposed have been developed. With this aim, the framework has been implemented and tested in a real customer support application currently run by a company. This company receives tickets about user problems that have to be solved by a group of technicians. The case-based argumentation framework presented in this thesis has been integrated as an argumentation module that agents that represent technicians can use to argue and persuade other agents to accept their proposed solutions as the best way to solve each problem.

To perform the tests we have used a 48 tickets case-base with real domain knowledge stored by a customer support company. The small size of this case-base has influenced the choice of several evaluation parameters, such as the number of agents to make the tests, the number of simulations and the weight for the similarity degree and the support factor. Also, in the company the group that the operators belong and their dependency relations are known by all technicians. Concretely, we have assumed that

all agents belong to the same group to allow them to populate their domain-case bases with random cases extracted from the same tickets case-base. We plan to extend the evaluation of our framework by updating the tickets case-base, as the company will provide us with new information, and by applying them to different domains, where different assumptions need to be make.

On one hand, the performance of the system has been evaluated under different settings. The tests show how those agents that follow an argumentation policy based on the proposals of this thesis are able to provide more accurate solutions to the problems that the system receives. The ability of the agents to argue allows those who have better arguments to support their decisions to win the argumentation dialogue. Therefore, the solutions with higher quality are selected among those proposed.

In terms of the percentage of agreement, the argumentative agents get better results than agents following other policies, specially when the number of technicians that are arguing to solve the problem increases. Also, the number of agents that agree is clearly higher when agents have the ability to argue. In this case, agents that have proposed less accurate solutions are persuaded to withdraw them and accept other agents' proposals, resulting in more agreements reached and more agents supporting the same final decisions. In addition, it has been demonstrated that if an agent uses its argumentation knowledge and has enough domain-knowledge, its positions are accepted by other agents.

The influence of the amount of argumentation knowledge that argumentative agents have has also been evaluated. If only one agent has argumentation knowledge that matches the context of current argumentation processes, as this knowledge increases, the amount of agreements reached and the number of agents that agree also increases. This demonstrates that this agent is effectively using its argumentation knowledge to select the best positions and arguments to put forward in a dialogue with other agents. Thus, as many useful arguments an agent has, as more proficient the agent is to persuade other agents to accept its proposals. However, if all or most agents have the ability of learning from argumentation dialogues, all of them have the same (high) persuasive power to defend their decisions and the agreement is difficult to achieve.

The different argumentation strategies (as combinations of an agent profile and different tactics) that an agent can follow to argue with other agents have been also compared. The tests performed evaluate the percentage of agreements reached and the percentage of agents of different profiles persuaded by an argumentative agent, which complies with

the reasoning process and dialogue protocol proposed for the case-based argumentation framework presented in this thesis. The results obtained demonstrate that if all participants of the argumentation process have an argumentative profile, more agreements are reached and more agents support the final solutions provided by the system. For all strategies, the best results are obtained when all agents are argumentative. Among them, the strategies that guide the agent that follows them to propose positions and arguments that have a higher potential probability of being accepted (ST2) and that minimise the number of potential attacks that an agent can receive (ST4) are the winning strategies. Future work will evaluate further combinations of profile-tactic for the initiator and also for the rest of the agents participating in the dialogue.

Finally, the influence of the knowledge that an agent has about the social context of their partners has been also evaluated. Results show that our defeat relation among arguments of agents with different dependency relations makes an expert's or a manager's arguments more important than the arguments of operators. Therefore, is an expert actually is better informed to assign better solutions to specific types of problems, the performance of the system improves. However, if a manager just has more amount of information than the operators, but this information do not make the manager to propose more accurate solutions, the performance of the system is adversely affected. The quantity of knowledge that agents has about the values of other agents also determines the good performance of the system. Therefore, if an agent assigns the same importance (weight) to both domain and argumentation knowledge to generate and select positions and arguments, but it does not know the value preferences of their partners, many times the agent uses argument-cases that are not suitable for the current situation. This makes the agent to make wrong decisions that worsen the global performance of the system.

We have assumed in this example that agents do their best to win the argumentation dialogue, thus following a persuasion dialogue, since in this way they get economical rewards and increase prestige. Despite that, those solutions that are better supported prevail. Hence, if agents do not follow a dialogue strategy that deviates the final outcome of the dialogue to fit their individual objectives, no matter if they give rise to the best solution to apply, the system reaches agreements that produce high quality solutions for the tickets received. This assumption has allowed us to perform more comprehensive tests with the small amount of data that we have and to check the advantages of following different dialogue strategies and of the amount of available knowledge about the preferences of other agents. However, a cooperative approach

where agents are not interested and collaborate to reach the best agreement would be appropriate for this example and will be implemented and evaluated in the future.

Part V

Conclusions

Chapter

7

Conclusions

This chapter summarises the main contributions of this PhD research and identifies future work to extend these contributions. The chapter also presents the list of publications where the main results of the PhD thesis have been presented.

7.1 Contributions

This PhD work presents an hybrid AI system that mixes different AI techniques and methods, such as case-based reasoning, semantic reasoning with ontologies, argumentation and multi-agent systems. Our main contribution consists in the proposal of a case-based argumentation framework that allows agents to argue in agent societies, taking into account their roles, preferences over values and dependency relations. The work developed in this thesis covers the objectives proposed in Chapter 1 at different levels.

On the *State of the Art Revision Level*, the thesis reviews the main concepts of argumentation theory that have been applied to AI and specifically, to MAS. The argumentation framework proposed in this research makes use of several of these concepts. Concretely it considers *argumentation schemes* as a knowledge resource that agents can use to generate and attack arguments. Also, the communication protocol between the agents

of the framework is modelled as a *dialogue game*, which uses *commitment stores* to track the positions and arguments that the agents engaged an argumentation dialogue have taken during it. As part of the state of the art revision, the challenges of the CBR methodology applied to argumentation in MAS have also been analysed. First, the few approaches that make this innovative integration are presented and discussed following different perspectives to compare them. Then, open research issues in this area are highlighted. These are: 1) the opportunities of argumentation to coordinate the interaction between agents that belong to a society; 2) the advantages to represent arguments in the form of cases; 3) the possible roles that the CBR methodology can play in the argumentation process; and 4) the advantages that agents can get by following different argumentation strategies. This PhD thesis deals with these open issues.

On the *Formal Level*, we have analysed the concept of agent society provided in the literature. From this analysis, we have concluded that in most cases the concept is quite ambiguous and is basically used as a synonym of a group a agents. However, we consider that the formal characterisation of an agent society should be enriched with extra features, such as the dependency relation that exists between the agents or the values that they want to promote with their decisions and actions. Our formal definition of agent society includes these features. Also, we have identified a set of desired requirements that an argumentation framework for agent societies should meet. These are: 1) be computationally tractable and designed to ease the performance of automatic reasoning processes over it; 2) be rich enough to represent general and context dependent knowledge about the domain and social information about the agents' dependency relations or the agents' group; 3) be generic enough to represent different types of arguments and 4) comply with the technological standards of data and argument interchange on the web. These requirements suggest that an argumentation framework for agent societies should be easily interpreted by machines and have highly expressive formal semantics to define complex concepts and relations over them. Thus, we have followed a knowledge-intensive case-based argumentation framework, where the case representation language is specified as an OWL 2 ontology. Most argumentation systems produce arguments by applying a set of inference rules. In open MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. However, tracking the arguments that agents put forward in argumentation processes could be relatively simple. Finally, we have provided an abstract formalisation for our argumentation framework for agent societies by extending ab-

stract value-based argumentation frameworks to work with agent societies. After that, we have also instantiated the framework by defining its elements.

On the *Agent Level*, our framework proposes individual knowledge resources for each agent to generate its positions and arguments. On one hand, agents have a domain-cases case-base, with domain-cases that represent previous problems and their solutions. On the other hand, agents have an argument-cases case-base, with argument-cases that represent previous argumentation experiences and their final outcome. In addition, agents can accede to a set of argumentation schemes, which represent stereotyped patterns of common reasoning in the application domain where the framework is implemented. All these resources are represented by using ontologies. Concretely, the framework uses a domain-dependent ontology to represent domain-cases and argumentation schemes and a generic argumentation ontology to represent argument-cases and arguments that the agents interchange. Thus, we have developed the case-based argumentation ontology called ArgCBROnto with this thesis.

The reasoning process that agents of our framework can use to manage positions and arguments has been also defined. Here, we have shown a series of generic algorithms which low level functions (e.g. concrete similarity measures, values of weights, etc.) can be instantiated depending on the application domain. By following this reasoning process, agents are able to generate positions and arguments, select the best positions and arguments to put forward in view of their argumentation experience, and evaluate other agents' proposals.

On the *System Level* we have proposed a dialogue game protocol to control the argumentation process between the agents of the framework. This protocol includes the definition of the locutions that agents can utter, the rules for making these utterances, the rules for maintaining the commitment store of each agent, the rules for speaker order and the commencement and termination rules. Furthermore, we have proposed a set of heuristic dialogue strategies that agents can use to improve the performance of their argumentation dialogues. Also at this level, we provide the formal axiomatic and operational semantics for the locutions of our dialogue game protocol. This semantics provides a common understanding about the properties of the communication language between agents.

Finally, on the *Evaluation Level* we have tested our proposals in two cases of study. On one hand, the formal specification of the framework has been applied to a water-right transfer domain where agents engage in argumentation dialogues to reach agreements

over the allocation of water resources. This is a theoretic example where the semantic properties of the framework have been validated. On the other hand, the framework has been implemented to develop an application on the customer support domain. Here, we consider a helpdesk of a call center where several operators must interact to solve jointly an incidence received by the center. In this domain several tests have been developed to analyse the performance of the framework in a real system. Concretely, we have tested the suitability of the solutions agreed by the operators, the influence of the argumentation strategies on the outcome and efficiency of the dialogue and the influence of different social context on the performance of the system.

7.2 Future Lines of Research

Due to the hybrid approach followed in this PhD thesis, there is a wide range of open issues for extending its contribution by advancing research in any of the research areas that is covers, such as argumentation, case-based reasoning, ontological reasoning and multi-agent systems. Here, we highlight those that we consider that are the most interesting from our point of view.

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

On the formalisation of multi-agent dialogues there are multiple possibilities that can be still investigated. One disadvantage of dialogue games is that they depend on the locutions of the game and, in most really implemented cases, these depend on the knowledge resources of the system. An alternative could be to develop a generic theory for modelling the way in which agents can argue in their societies based on the work presented in [Perelman and Olbrechts-Tyteca, 1969] about the arguments based on the structure of reality. There, several stereotyped patterns of the way that humans argue by taking into account their belonging to a society were analysed.

Also, along this thesis we have avoided to deal with trust and reputation issues that

can compromise the veracity of the beliefs and utterances of agents. In real open MAS, malicious agents could communicate false knowledge or try to prevent the reach of an agreement on purpose. The literature on trust and reputation methods for MAS is extensive. This paves the way for the adaption of some of well-known methods for the particular setting of a society of agents that argue by using argumentation techniques.

For clarity purposes, we have assume in this thesis that agents are in the same groups to participate in an argumentation dialogue with other agents. However, agents can belong to several groups. Nevertheless, in real scenarios agents from different groups that have different preferences over values can engage in the same conversation. In this case, it is unlikely that the agent of a group knows the preferences of the other group. The same happens when agents of the same group do not know the social features of their fellows due to restrictive privacy policies. Thus, new algorithms to infer unknown values about the social context of the opponent have to be developed.

Depending on the application domain, the preferences over values of each agent or group, the same values or even the dependency relations between the agents of the framework could change with the curse of the time. Again, this requires for new adaption methods that allow the agent society to reconfigure its social parameters dynamically. In addition, the concrete algorithms used in this thesis for performing each phase of the CBR cycle (retrieve, reuse, revise and retain) have been selected for their simplicity and suitability to the customer support application domain. The development of new and efficient learning algorithms was not in our research objectives. However, a more comprehensive analysis could compare the performance of these algorithms with different approaches, since the work performed by the CBR community to devise new methods to implement the CBR phases produce interesting and better results continuously.

Finally, in this work we do not have considered temporal restrictions in the agents' reasoning process and in the argumentative dialogues that they have. However, real application scenarios can impose these temporal bounds to reach agreements. In fact, for some systems, like route planing or medical diagnosis, to reach an agreement before a deadline can be crucial. As the next step for our research, we plan to investigate how to adapt the case-based argumentation framework proposed in this thesis to be used in real time application scenarios.

7.3 Related Publications

This section presents the publications associated with the PhD thesis that have been published to date. They are organised in chronological order and separated by the type of the publication

Journals

- S. Heras, V. Botti and V. Julián. Argument-based Agreements in Agent Societies. Neurocomputing pp. In Press. **JCR 1.440 - Q2**.
- M. Navarro, S. Heras, V. Julián and V. Botti. Incorporating Temporal-Bounded CBR techniques in Real-Time Agents. Expert Systems with Applications Vol. 38 N. 3 pp. 2783-2796. Elsevier (2011) **JCR 2.908 - Q1**.
- S. Heras, V. Julián and V. Botti. Applying CBR to Manage Argumentation in MAS. International Journal of Reasoning-based Intelligent Systems Vol. 2 N. 2 pp. 110-117. Inderscience Publishers (2010)
- S. Heras, V. Botti and V. Julián. Challenges for a CBR Framework for Argumentation in open MAS. Knowledge Engineering Review Vol. 24 N. 4 pp. 327-352. Cambridge University Press (2009) **JCR 1.143 - Q3**
- S. Heras, J. A. García-Pardo, R. Ramos-Garijo, A. Palomares, V. Botti, M. Rebollo and V. Julián. Multi-domain case-based module for customer support. Expert Systems with Applications Vol. 36 N. 3 pp. 6866-6873. Elsevier (2009) **JCR 2.908 - Q1**.
- S. Heras, M. Navarro and V. Julián. Hybrid Reasoning and Coordination Methods on Multi-Agent Systems. Journal of Physical Agents Vol. 3 N. 3 pp. 1-2. (2009)
- S. Heras, N. Criado, E. Argente and V. Julián. Norm Emergency through Argumentation. Journal of Physical Agents Vol. 3 N. 3 pp. 31-38. (2009)

Conferences and Workshops

- J. Jordán, S. Heras and V. Julián. A Customer Support Application Using Argumentation in Multi-Agent Systems. 14th International Conference on Information

Fusion (FUSION-11) pp. In Press. International Society of Information Fusion (ISIF) **CORE C**

- J. Jordán, S. Heras, S. Valero and V. Julián. An Argumentation Framework for Supporting Agreements in Agent Societies Applied to Customer Support 6th International Conference on Hybrid Artificial Intelligence Systems (HAIS-11) pp. In Press. LNAI, Springer **CORE C**

- S. Heras, K. Atkinson, V. Botti, F. Grasso, V. Julian and P. McBurney. How Argumentation can Enhance Dialogues in Social Networks. 3rd International Conference on Computational Models of Argument (COMMA-10) Vol. 216 pp. 267-274. IOS Press (2010)

- S. Heras, K. Atkinson, V. Botti, F. Grasso, V. Julian and P. McBurney. Applying Argumentation to Enhance Dialogues in Social Networks. ECAI 2010 workshop on Computational Models of Natural Argument (CMNA-10) pp. 10-17. (2010)

- S. Heras, M. Navarro, V. Botti and V. Julián. Applying Dialogue Games to Manage Recommendation in Social Networks. Argumentation in Multi-Agent Systems, LNAI Vol. 6057 pp. 256-272. Springer (2010)

- S. Heras, V. Botti and V. Julián. An Abstract Argumentation Framework for Supporting Agreements in Agent Societies. 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS-10) LNAI Vol. 6077 N. 2 pp. 177-184. Springer (2010) **CORE C**

- S. Heras. Strategic Argumentation in Open Multi-Agent Societies. 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10) pp. 1663-1664. ACM Press (2010) **CORE A**

- S. Heras, V. Botti and V. Julián. On a Computational Model of Argument for Agent Societies. 7th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS-10) at (AAMAS-10) pp. 55-72. ACM Press (2010)

- N. Criado, S. Heras, E. Argente and V. Julián. Normative Argumentation. 8th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS). Advances in Soft Computing Vol. 71 pp. 29-36. Springer (2010)

- N. Criado, S. Heras, E. Argente and V. Julián. Contract Argumentation in Virtual Organizations. 5th International Workshop on Normative Multi-Agent Systems, (NORMAS-10) pp. 55-59. (2010)

- S. Heras, M. Navarro, V. Botti and V. Julián. Applying Dialogue Games to Manage Recommendation in Social Networks. 7th European Workshop on Multi-Agent Systems, EUMAS-09 pp. 1-15. (2009) **CORE C**

- S. Heras, M. Navarro, V. Botti and V. Julián. Applying Dialogue Games to Manage Recommendation in Social Networks. AAMAS 6th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS-09) pp. 55-70. ACM Press (2009)

- S. Heras, N. Criado, E. Argente and V. Julián. A Dialogue-Game Approach for Norm-based MAS Coordination. 4th International Conference on Hybrid Artificial Intelligence Systems (HAIS-09) LNAI Vol. 5572 pp. 468-475. Springer (2009) **CORE C**

- S. Heras, M. Rebollo and V. Julián. Arguing About Recommendations in Social Networks. Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology pp. 314-317. IEEE Press (2008) **CORE B**

- S. Heras, M. Rebollo and V. Julián. A Dialogue Game Protocol for Recommendation in Social Networks. 3rd International Workshop on Hybrid Artificial Intelligence Systems (HAIS-08). LNAI Vol. 5271 pp. 515-522. Springer (2008) **CORE C**

- S. Heras, V. Julián and V. Botti. CBR Contributions to Argumentation in MAS. 2nd International Workshop on Hybrid Artificial Intelligence Systems (HAIS-07). Advances in Soft Computing Vol. 44 pp. 304-311. Springer (2007) **CORE C**

- S. Heras, J. A. García-Pardo, M. Rebollo, V. Julián and V. Botti. Intelligent Customer Support for Help-Desk Environments. Hybrid Artificial Intelligence Systems pp. 73-80. Universidad de Salamanca (2007)

- S. Heras, J. A. García-Pardo, R. Ramos-Garijo, A. Palomares, V. Julián, M. Rebollo and V. Botti. CBR Model for the Intelligent Management of Customer Support Centers. 7th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-06) Vol. LNCS 4224 pp. 663-670. Springer (2006) **CORE C**

- J. A. García-Pardo, S. Heras, R. Ramos-Garijo, A. Palomares, V. Julián, M. Rebollo and V. Botti. CBR-TM: A New Case-Based Reasoning System for Help-Desk Environments. 17th European Conference on Artificial Intelligence (ECAI-06) Vol. 141 pp. 833-834. IOS Press (2006) **CORE A**

Bibliography

Aamodt, A. (2004). Knowledge-intensive case-based reasoning in creek. In *7th European Conference on Case-Based Reasoning, ECCBR-04*, pages 1–15. Springer.

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

Aleven, V. and Ashley, K. D. (1997). Teaching case-based argumentation through a model and examples, empirical evaluation of an intelligent learning environment. In *Artificial Intelligence in Education, AIED-97*, volume 39 of *Frontiers in Artificial Intelligence and Applications*, pages 87–94. IOS Press.

Amgoud, L. (2003). A formal framework for handling conflicting desires. In *7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU-03*, volume 2711 of *LNAI*, pages 552–563. Springer.

Amgoud, L. and Hameurlain, N. (2006). A formal model for designing dialogue strategies. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-06*, pages 414–416. ACM Press.

Amgoud, L. and Kaci, S. (2004). On the generation of bipolar goals in argumentation-based negotiation. In *In Argumentation in Multi-Agent Systems: State of the art survey*, volume 3366 of *LNAI*, pages 192–207. Springer.

Amgoud, L. and Kaci, S. (2005). On the study of negotiation strategies. In *AAMAS 2005 Workshop on Agent Communication, AC-05*, pages 3–16. ACM Press.

Amgoud, L. and Maudet, N. (2002). Strategical considerations for argumentative agents

(preliminary report). In *9th International Workshop on Non-Monotonic Reasoning, NMR-02*, LNAI, pages 399–407. Springer.

Amgoud, L., Maudet, N., and Parsons, S. (2000). Modelling dialogues using argumentation. In *4th International Conference on MultiAgent Systems, ICMAS-00*. IEEE Press.

Amgoud, L. and Parsons, S. (2001). Agent dialogues with conflicting preferences. In *5th International Workshop on Agent Theories, Architectures and Languages, ATAL-01*, LNAI, pages 1–17. Springer.

Armengol, E. and Plaza, E. (2001). Lazy induction of descriptions for relational case-based learning. In *12th European Conference on Machine Learning, ECML-01*, pages 13–24. Springer-Verlag.

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

Ashley, K. D. (1991). Reasoning with cases and hypotheticals in hypo. *International Journal of Man-Machine Studies*, 34:753–796.

Atkinson, K. (2005a). A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems. Special issue on Argumentation in Multi-Agent Systems*, 11(2):153–171.

Atkinson, K. (2005b). *What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning*. PhD thesis, Liverpool University.

Aulinas, M., Tolchinsky, P., Turon, C., Poch, M., and Cortés, U. (2007). Is my spill environmentally safe? towards an integrated management of wastewater in a river basin using agents that can argue. In *7th International IWA Symposium on Systems Analysis and Integrated Assessment in Water Management, WATERMATEX-07*.

Baader, F., Horrocks, I., and Sattler, U. (2007). *Handbook of Knowledge Representation*, chapter Description Logics, pages 135–179. Elsevier.

Baroni, P. and Giacomin, M. (2009). *Argumentation in Artificial Intelligence*, chapter Semantics of Abstract Argument Systems, pages 25–44. Springer.

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

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

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

Bench-Capon, T. J. (1989). Deep models, normative reasoning and legal expert systems. In *2nd iInternational Conference on Artificial Intelligence and Law, ICAIL-89*, pages 37–45. ACM Press.

Bench-Capon, T. J. (1998). Specification and implementation of toulmin dialogue game. In *International Conferences on Legal Knowledge and Information Systems, JURIX-98*, Frontiers of Artificial Intelligence and Applications, pages 5–20. IOS Press.

Bench-Capon, T. J. and Visser, P. (1997). Ontologies in legal information systems: The need for explicit specifications of domain conceptualisations. In *6th International Conference on Artificial intelligence and Law, ICAIL-97*, pages 132–141. ACM Press.

Botti, V., Garrido, A., Gimeno, J. A., Giret, A., Igual, F., and Noriega, P. (2010). An Electronic Institution for Simulating Water-Right Markets. In *3rd Workshop on Agreement Technologies, WAT-10*, pages 3–18.

Botti, V., Garrido, A., Giret, A., Igual, F., and 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., and 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*, volume 494 of *LNCS*, pages 1–10. Springer.

Branting, L. K. (1991). Building explanations from rules and structured cases. *International Journal of Man-Machine Studies*, 34(6):797–837.

Brüninghaus, S. and Ashley, K. D. (2001). Improving the representation of legal case texts with information extraction methods. In *8th International Conference on Artificial Intelligence and Law, ICAIL-01*, pages 42–51. ACM Press.

Brüninghaus, S. and Ashley, K. D. (2003). Predicting the outcome of case-based legal arguments. In *9th International Conference on Artificial Intelligence and Law, ICAIL-03*, pages 233–242. ACM Press.

Brüninghaus, S. and Ashley, K. D. (2005). Generating legal arguments and predictions from case texts. In *10th International Conference on Artificial Intelligence and Law, ICAIL-05*, pages 65–74. ACM Press.

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

Capobianco, M., nevar, C. I. C., and Simari, G. R. (2005). Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems*, 11(2):127–151.

Chesñevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., and Willmott, S. (2006). Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316.

Criado, N., Argente, E., and Botti, V. (2009). A Normative Model For Open Agent Organizations. In *International Conference on Artificial Intelligence, ICAI-09*. CSREA Press.

Daniels, J. J. and Rissland, E. L. (1997). Finding legally relevant passages in case opinions. In *6th International Conference on Artificial Intelligence and Law, ICAIL-97*, pages 39–47. ACM Press.

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

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

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

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

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

Fox, J. and Parsons, S. (1998). Arguing about beliefs and actions. In *Applications of Uncertainty Formalisms*, volume 1455, pages 266–302. Springer.

Garrido, A., Giret, A., and Noriega, P. (2009). mWater: a Sandbox for Agreement Technologies. In *12th International Congress of the Catalan Association of Artificial Intelligence - CCIA-09*, volume 202, pages 252–261. IOS Press.

Giret, A., Garrido, A., and Botti, V. (2010). D8.2.1 Report. mWater. Technical Report AT/2008/D8.2.1/v0.1, Universidad Politecnica de Valencia.

Gordon, T. F. (1987). Oblog-2: A hybrid knowledge representation system for defeasible reasoning. In *1st International Conference on Artificial Intelligence and Law, ICAIL-87*, pages 231–39. ACM Press.

Gordon, T. F. and Karacapilidis, N. (1997). The zeno argumentation framework. In *6th International Conference on Artificial Intelligence and Law, ICAIL-97*, pages 10–18. ACM Press.

Hage, J. and Verheij, B. (1995). Reason-based logic: a logic for reasoning with rules and reasons. *Law, Computers and Artificial Intelligence*, 3(2-3):171–209.

Hamblin, C. L. (1970). *Fallacies*. Methuen Co. Ltd.

Heras, S., Atkinson, K., Botti, V., Grasso, F., Julián, V., and McBurney, P. (2009a). How Argumentation can Enhance Dialogues in Social Networks. Technical Report ULCS-09-020, University of Liverpool, UK.

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

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

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

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. and Patel-Schneider, P. (2004). Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics*, 1(4):345–357.

Hulstijn, J. (2000). *Dialogue Models for Inquiry and Transaction*. PhD thesis, University of Twente.

Hulstijn, J. and van der Torre, L. (2004). Combining goal generation and planning in an argumentation framework. In *10th International Workshop on Non-Monotonic Reasoning, NMR-04*, LNAI. Springer-Verlag.

Jakobovits, H. and Vermeir, D. (1999). Dialectic semantics for argumentation frameworks. In *7th International Conference on Artificial Intelligence and Law, ICAIL-99*, pages 53–62. ACM Press.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C., and Wooldridge, M. (2001). Automated negotiation: prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215.

Kakas, A., Maudet, N., and Moraitis, P. (2005). Modular Representation of Agent Interaction Rules through Argumentation. *Autonomous Agents and Multi-Agent Systems*, 11:189–206.

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

Karacapilidis, N., Trousse, B., and Papadias, D. (1997). Using case-based reasoning for argumentation with multiple viewpoints. In *2nd International Conference on Case-Based Reasoning Research and Development, ICCBR-97*, pages 541–552. Springer.

Karlins, M. and Abelson, H. I. (1970). *Persuasion: How Opinions and Attitudes are Changed*. Springer.

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

Kripke, S. (1959). A completeness proof in modal logic. *Journal of Symbolic Logic*, 24:1–14.

Lodder, A. and Herczog, A. (1995). DIALAW - a dialogical framework for modelling legal reasoning. In *5th International Conference on Artificial Intelligence and Law, ICAIL-95*, pages 146–155. ACM Press.

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

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

MacKenzie, J. D. (1978). Question-begging in non-cumulative systems. *Philosophical Logic*, 8(1):117–133.

Maudet, N. and Chaib-draa, B. (2002). Commitment-based and dialogue-game based protocols-news trends in agent communication language. *Knowledge Engineering Review*, 17(2):157–179.

Maudet, N. and Evrard, F. (1998). A generic framework for dialogue game implementation. In *2nd Workshop on Formal Semantics and Pragmatics of Dialogue*, pages 185–198. University of Twente.

McBurney, P., Eijk, R. M. V., Parsons, S., and Amgoud, L. (2003). A dialogue game protocol for agent purchase negotiations. *Autonomous Agents and Multi-Agent Systems*, 7(3):235–273.

McBurney, P., Hitchcock, D., and Parsons, S. (2007). The eightfold way of deliberation dialogue. *International Journal of Intelligent Systems*, 22(1):95–132.

McBurney, P. and Parsons, S. (2001). Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence, Special Issue on Representations of Uncertainty*, 32(1-4):125–169.

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

McBurney, P. and Parsons, S. (2002b). Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11(3):315–334.

McBurney, P. and Parsons, S. (2004). Locutions for argumentation in agent interaction protocols. In *Revised Proceedings of the International Workshop on Agent Communication, AC-04*, volume 3396 of *LNAI*, pages 209–225. Springer.

McBurney, P. and Parsons, S. (2009). *Argumentation in Artificial Intelligence*, chapter Dialogue games for agent argumentation, pages 261–280. Springer.

Modgil, S., Tolchinsky, P., and Cortés, U. (2005). Towards formalising agent argumentation over the viability of human organs for transplantation. In *4th Mexican International Conference on Artificial Intelligence, MICAI-05*, LNAI, pages 928–938. Springer-Verlag.

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

Ontañón, S. and Plaza, E. (2006). Arguments and counterexamples in case-based joint deliberation. In *3rd International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-06*. ACM Press.

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

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

Ossowski, S., Julián, V., Bajo, J., Billhardt, H., Botti, V., and Corchado, J. M. (2007). Open issues in open mas: An abstract architecture proposal. In *12 Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA-07*, volume 2 of *LNCS*, pages 151–160. 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.

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

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

Prakken, H. (1993). *Logical Tools for Modelling Legal Arguments*. PhD thesis, Free University Amsterdam.

Prakken, H. (2006). Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 21:163–188.

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

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

Prakken, H. and Sartor, G. (1998). Modelling reasoning with precedents in a formal dialogue game. *Artificial Intelligence and Law*, 6:231–287.

Rahwan, I. (2006). Argumentation in multi-agent systems. *Autonomous Agents and Multiagent Systems, Guest Editorial*, 11(2):115–125.

Rahwan, I. and Amgoud, L. (2006). An argumentation-based approach for practical reasoning. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-06*, pages 347–354. ACM Press.

Rahwan, I. and Larson, K. (2008). Mechanism design for abstract argumentation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, volume 2, pages 1031–1038. ACM Press.

Rahwan, I. and Larson, K. (2009). Argumentation and Game Theory. *Argumentation in Artificial Intelligence*, pages 321–339.

Rahwan, I., Larson, K., and Tohmé, F. (2009). A characterisation of strategy-proofness for grounded argumentation semantics. In *21st International Jont Conference on Artifical Intelligence, IJCAI-09*, pages 251–256. Morgan Kaufmann Publishers Inc.

Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons, S., and Sonenberg, L. (2003). Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375.

Rahwan, I. and Simari, G., editors (2009). *Argumentation in Artificial Intelligence*. Springer.

Rahwan, I., Sonenberg, L., Jennings, N. R., and McBurney, P. (2007a). STRATUM: A methodology for designing heuristic agent negotiation strategies. *Applied Artificial Intelligence*, 21(6):489–527.

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

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

Reed, C. and Walton, D. (2005). Towards a formal and implemented model of argumentation schemes in agent communication. *Autonomous Agents and Multi-Agent Systems*, 11(2):173–188.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13:81–132.

Rissland, E. L., Ashley, K. D., and Branting, L. K. (2006). Case-based reasoning and law. *The Knowledge Engineering Review*, 20(3):293–298.

Rissland, E. L., Ashley, K. D., and Loui, R. (2003). AI and Law: A fruitful synergy. *Artificial Intelligence*, 150(1-2):1–15.

Rissland, E. L. and Skalak, D. B. (1991). CABARET: Rule interpretation in a hybrid architecture. *International Journal of Man-Machine Studies*, 34:839–887.

Rissland, E. L., Skalak, D. B., and Friedman, M. T. (1993). BankXX: a program to generate argument through case-based search. In *4th International Conference on Artificial Intelligence and Law, ICAIL-93*, pages 117–124. ACM Press.

Rittel, H. and Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4:155–169.

Roth, B. and Rotolo, A. (2007). Strategic argumentation: a game theoretical investigation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pages 81–90. ACM Press.

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

Sadri, F., Toni, F., and Torroni, P. (2001a). Dialogues for negotiation: Agent varieties and dialogue sequences. In *Revised Papers from the 8th International Workshop on Intelligent Agents VIII, ATAL-01*, volume 2333, pages 405–421. Springer.

Sadri, F., Toni, F., and Torroni, P. (2001b). Logic agents, dialogue, negotiation - an abductive approach. In *Convention of The Society for the Study of Artificial Intelligence and the Simulation of Behaviour, AISB-01*. AISB.

Schneider, S., editor (1996). *Encyclopedia of Climate and Weather*. Oxford University Press.

Searle, J. (2001). *Rationality in Action*. MIT Press.

Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press.

Simari, G. R., Garcia, A. J., and Capobianco, M. (2004). Actions, planning and defeasible reasoning. In *10th International Workshop on Non-Monotonic Reasoning, NMR-04*, LNAI, pages 377–384. Springer.

Singh, M. (2000). A social semantics for agent communication languages. volume 1916 of *LNCS*, pages 31–45. Springer.

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

Soh, L.-K. and Tsatsoulis, C. (2001a). Agent-based argumentative negotiations with case-based reasoning. In *Working Notes of the AAAI Fall Symposium Series on Negotiation Methods for Autonomous Cooperative Systems*, pages 16–25. AAAI Press.

Soh, L.-K. and Tsatsoulis, C. (2001b). Reflective negotiating agents for real-time multisensor target tracking. In *17th International Joint Conference on Artificial In-*

telligence, IJCAI-01, volume 2, pages 1121–1127. Morgan Kaufmann Publishers Inc.

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

Sørmo, F., Cassens, J., and Aamodt, A. (2005). Explanation in case-based reasoning - perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143.

Sycara, K. (1987). *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods*. PhD thesis, Georgia Institute of Technology.

Sycara, K. (1989). Argumentation: Planning other agents' plans. In *11th International Joint Conference on Artificial Intelligence, IJCAI-89*, volume 1, pages 517–523. Morgan Kaufmann Publishers Inc.

Sycara, K. (1990). Persuasive argumentation in negotiation. *Theory and Decision*, 28:203–242.

Tennent, R. D. (1991). *Semantics of Programming Languages*. Prentice Hall.

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

Tolchinsky, P., Cortés, U., Modgil, S., Caballero, F., and López-Navidad, A. (2006a). Increasing human-organ transplant availability: Argumentation-based agent deliberation. *IEEE Intelligent Systems*, 21(6):30–37.

Tolchinsky, P., Modgil, S., and Cortés, U. (2006b). Argument schemes and critical questions for heterogeneous agents to argue over the viability of a human organ. In *AAAI 2006 Spring Symposium Series; Argumentation for Consumers of Healthcare*. AAAI Press.

Tolchinsky, P., Modgil, S., Cortés, U., and Sànchez-Marrè, M. (2006c). CBR and argument schemes for collaborative decision making. In *1st International Conference on Computational Models of Argument, COMMA-06*, volume 144, pages 71–82. IOS Press.

Toulmin, S. E. (1958). *The Uses of Argument*. Cambridge University Press.

van Eemeren, F. H. and Grootendorst, R. (1984). *Speech acts in argumentative discussions*. Foris Publications.

van Eemeren, F. H. and Grootendorst, R. (1992). *Argumentation, communication, and fallacies: a pragma-dialectical perspective*. Routledge Publishers.

van Eemeren, F. H. and Grootendorst, R. (2004). *A Systematic Theory of Argumentation: The pragma-dialectical approach*. Cambridge University Press.

van Eijk, R. M. (2002). Semantics of Agent Communication: An Introduction. In *Foundations and Applications of Multi-Agent Systems, UKMAS 1996-2000, Selected Papers*, volume 2403 of *LNAI*, pages 152–168. Springer-Verlag.

Vázquez-Salceda, J., Cortés, U., Padget, J., López-Navidad, A., and Caballero, F. (2003). The organ allocation process: A natural extension of the carrel agent-mediated electronic institution. *AI Communications*, 16(3):153–165.

Walton, D. (1996). *Argumentation Schemes for Presumptive Reasoning*. Routledge Publishers.

Walton, D. and Krabbe, E. C. W. (1995). *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press.

Walton, D., Reed, C., and Macagno, F. (2008). *Argumentation Schemes*. Cambridge University Press.

Wardeh, M., Bench-Capon, T., and 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*, pages 133–146. Springer.

Watson, I. (1997). *Applying case-based reasoning. Techniques for enterprise systems*. Morgan Kaufmann Pusblishers, Inc.

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