Universitat de Girona

Departamento de Electrónica, Informática y Automática

PhD Thesis

On Intentional and Social Agents with Graded Attitudes.

Dissertation submitted by:

Ana Casali

Supervisors:
Lluís Godo and Carles Sierra

Institut d'Investigació en Intel·ligència Artificial
IIIA - CSIC

November 2008

To Jorge, in our 25th anniversary.

For it is God who works in you to will and to act according to his good purpose

Phil 3:13, The Bible

Acknowledgements

I must start by expressing my gratitude to God for giving me the will to face up this challenge and the strength to achieve it. This dissertation would not have been possible without the personal and professional support offered by many people. I would like to thank my parents Carlos and Ena for teaching me by their examples the culture of work and of perseverance until success is reached. This thesis would not have been achieved either without the love, support and patience of my husband Jorge and our four children: Gerardo, Verónica, Karen and Brenda. They have stayed with me side by side during this period of hard work, helping me and bringing me lovely care. I love them very much.

I would like to express my deep gratitude to both supervisors Dr. Lluís Godo and Dr. Carles Sierra, whose guide have proved really helpful in this research work. Living thousands of kilometers away has made the work more difficult and the time they have devoted to this project has been very valuable. I also appreciate their effort to review all my work, give me their advice and improve my English performance. I have enjoyed working with them and I really thank them for offering me their support, enthusiasm and friendship.

I am also grateful to all the people of the Institut d'Investigació en Intel·ligència Artificial, for welcoming me in Bellaterra and offering me support during my stay in Spain. I have also received helpful comments from them which have encouraged me to carry out this research. I would like to thank Dr. Beatriz Lopez, professor at Universitat de Girona for assisting me in the first stages of this Doctoral Programme and for offering me her generous help. I am particularly grateful to my brother Gerardo and my Catalonian friends David and Montse for their warm hospitality.

My very especial gratitude also goes to all the community of the University Departments –Depto. de Sistemas e Informática and Depto. de Ciencias de la Computación, FCEIA, Universidad Nacional de Rosario, Argentina– where I have been working as a professor for more than twenty years. I am thankful to my dear companions who have made my working time more pleasant and who have made me overcome difficult working conditions. I would especially like to thank my statistician friends Marta and José Alberto Pagura for their expert advice in the Data Analysis, and my students Armando, Silvana, Eric and Iván for their support in the prototype implementation and experimentation. Moreover, I am grateful to

all those young people from my Departments for their enthusiasm and willingness to learn as well as to improve themselves. Their moving spirit has been a really great motivation for me.

Thanks to all!

Ana

Rosario, July 2008.

On Intentional and Social Agents with Graded Attitudes.

Abstract

The central contribution of this dissertation is the proposal of a graded BDI agent model (g-BDI), specifying an architecture capable of representing and reasoning with graded mental attitudes. We consider that making the BDI architecture more flexible will allow us to design and develop agents capable of improved performance in uncertain and dynamic environments, serving other agents (human or not) that may have a set of graded motivations.

In the g-BDI model, the agent graded attitudes have an explicit and suitable representation. Belief degrees represent the extent to which the agent believes a formula to be true. Degrees of positive or negative desires allow the agent to set different levels of preference or rejection respectively. Intention degrees also give a preference measure but, in this case, modelling the cost/benefit trade off of achieving an agent's goal. Then, agents having different kinds of behaviour can be modelled on the basis of the representation and interaction of their graded beliefs, desires and intentions. The formalization of the g-BDI agent model is based on Multi-context systems (MCS) and in order to represent and reason about the beliefs, desires and intentions, we followed a many-valued modal approach. Also, a sound and complete axiomatics for representing each graded attitude is proposed.

Besides, in order to cope with the operational semantics aspects of the g-BDI agent model, we first defined a Multi-context calculus (MCC) for Multi-context systems (MCS) execution. Through MCC we give this agent model computational meaning and in this way, we move one step closer to the development of an interpreter of the g-BDI agents.

Furthermore, a software engineering process to develop graded BDI agents in a multiagent scenario is presented. The aim of the proposed methodology is to guide the design of a multiagent system starting from a real world problem. Through the design and implementation of a Tourism recommender system, where one of its principal agents is modelled as a g-BDI agent, we show that the agent model is useful to design and implement concrete agents in real world applications.

Finally, using the case study we have made some experiments concerning the flexibility and performance of the g-BDI agent model, demonstrating that this agent model is useful to develop agents showing varied and rich behaviours. We also

show that the results obtained by these particular recommender agents using graded attitudes improve those achieved by agents using non-graded attitudes.

Contents

Contents **ix**

List of Figures **xiii**

List of Tables **xv**

I Introductory Concepts **1**

1 Introduction **3**
1.1 Motivations 4
1.2 Contributions 5
1.3 Structure 9

2 Related Work **11**
2.1 Introduction 11
2.2 Agent Theories and Architectures 11
2.2.1 Logic-Based Architectures - *Deductive agents* 13
2.2.2 Reactive architectures - *Reactive agents* 14
2.2.3 Layered Architectures - *Hybrid agents* 15
2.2.4 Practical Reasoning Architectures - *BDI agents* 17
2.2.5 Rao and Georgeff's BDI model 22
2.2.6 Advantages of BDI models 30
2.3 Multi-context Systems 31
2.3.1 Formalization of multi-context systems 31
2.3.2 Multi-context agents 33
2.3.3 Advantages of the multi-context specification of agents 35
2.4 Logics of preference 36
2.4.1 Bipolar representation of preferences 38
2.5 Graded Attitudes in Intentional Agent Architectures 42
2.5.1 Argumentation-based approaches to BDI agents 45
2.6 Conclusions 50

3 Logical Background **53**

3.1 Propositional Dynamic logic 53
3.2 $G_\Delta(C)$ and RPL Fuzzy Logics 55
3.2.1 Rational Lukasiewicz Logic 58

II The Graded BDI Agent Model 59

4 The General Framework 61
4.1 Introduction 61
4.2 Multi-context specification 62
4.3 Logical Framework: Many-valued modal approach 63

5 The Belief Context 67
5.1 Introduction 67
5.2 Belief context logics 68
5.3 The BC_{nec} Logic 70
5.4 The BC_{prob} Logic 74
5.5 Conclusions 76

6 Desire and Intention Contexts 79
6.1 Introduction 79
6.2 Desire Context (DC) 81
6.2.1 DC Language 81
6.2.2 Semantics for DC 82
6.2.3 DC Axioms and Rules 83
6.2.4 Consistency Schemas 87
6.3 Intention Context 91
6.3.1 IC Language 91
6.3.2 Semantics and axiomatization for IC 93
6.4 Conclusions 95

7 Functional contexts and Bridge rules 97
7.1 Planner and Communication Contexts 97
7.2 Bridge Rules 98
7.3 How the g-BDI model works 101
7.4 Example 103
7.5 Comparison with Rahwan and Amgoud's approach 105

8 The Socialization of the g-BDI agents 107
8.1 Introduction 107
8.2 Reasoning about other agent attitudes 107
8.3 Trust in an agent society 109
8.3.1 A Social Context to Filter Information 110
8.3.2 Trust in Delegation 112

8.3.3 Conclusions 116

9 Operational semantics for g-BDI Agents 117
9.1 Introduction 117
9.2 Mobile Ambient Calculus 118
9.3 Multi-context Calculus 121
9.4 Operational Semantics 125
9.5 Mapping a g-BDI Agent to the MCC 127
9.5.1 Mapping the Desire context into a Desire ambient 129
9.6 Conclusions 133

III Methodology and a Case-Study 135

10 Case study domain 137
10.1 Introduction 137
10.2 Recommender Agents 138
10.3 Recommender Systems in Travel and Tourism 141
10.4 Case Study: Recommender System on Argentinian Tourism 143

11 Methodology 147
11.1 Introduction 147
11.2 The Development Process of g-BDI Agent-Based Systems 149
11.3 System Analysis and Design Phase 153
11.3.1 Stage I: System Specification and Analysis 153
11.3.2 Stage II: System Architecture Design 156
11.4 Agent Design Phase 161
11.4.1 Stage III: A graded BDI Agent Design 162
11.4.2 Stage IV: Agent Detailed Design 164
11.5 Conclusions 170

12 Implementation 171
12.1 Introduction 171
12.2 Multiagent development 172
12.3 *T-Agent* Implementation 173
12.3.1 Communication Context 174
12.3.2 Desire Context 177
12.4 Belief Context 179
12.4.1 Tourist packages 179
12.4.2 Destination ontology 180
12.4.3 Special Relations in the domain 181
12.4.4 Beliefs on desires fulfillment 182

12.5 Planner Context . . . 184
12.6 Intention Context . . . 184
12.6.1 Estimating the expected satisfaction of desires . . . 185
12.6.2 Computing the intention degrees . . . 188
12.7 Conclusions . . . 189

IV Experimentation and Discussions 191

13 Experimentation 193
13.1 Introduction . . . 193
13.2 Validation . . . 194
13.3 Experimentation . . . 197
13.3.1 Sensitive model analysis . . . 198
13.3.2 Graded vs. non-graded model comparison . . . 201
13.4 Data analysis . . . 205
13.5 Conclusions . . . 208

14 Discussion 209
14.1 Contributions of this Thesis . . . 209
14.1.1 Contributions respect to BDI architectures . . . 209
14.1.2 Contributions on related fields . . . 211
14.2 Future work . . . 213
14.3 Related publications . . . 214

Bibliography 217

List of Figures

1.1 Related work to the g-BDI agent model development 6
1.2 Giving Operational Semantics to the g-BDI agent model 8
1.3 Software Engineering for the g-BDI agents 8

2.1 Information and control flows in layered agents architecture (Source: [111], p263). 16
2.2 Schematic diagram of a generic belief-desire-intention architecture (Source: [152], p58). 20
2.3 The procedural Reasoning System 21
2.4 Belief-accessible world (Source: [123]). 23
2.5 Compatibility between Belief and Goal-accessible world (Source: [123]). 24
2.6 Belief-accessible world relation 27
2.7 Different types of BDI agent. From left to right, the relations between modalities correspond to strong realism, realism and weak realism. (Source [115], p272). 34

4.1 Multi-context model of a graded BDI agent 63

7.1 Agent architecture 102

8.1 A g-BDI agent model including a SC for filtering information. 112
8.2 A g-BDI agent model including a SC to deal with delegation. 114

9.1 General structure of an ambient 120
9.2 The general ambient structure in MCC 122
9.3 The ambient structure for a g-BDI agent 129
9.4 The DC ambient structure 130

10.1 Principal items in a recommender schema 139
10.2 Structural view of tourism market 142
10.3 Case study in the tourism market 144
10.4 Recommender System on Argentinian Packages 145

11.1 Development Process for BDI Agent-Based Systems 151

11.2 Goal-Task Overview Diagram 158
11.3 Role interaction model 160
11.4 The multi-agent Recommender System 162
11.5 Detailed modelling process flow 165
11.6 Multi-context model of the graded BDI *T-Agent* 170

12.1 Multithread system scheme 173
12.2 User interface: tourist's preferences. 175
12.3 User interface: package recommendation, a package description and user feedback. 176

13.1 The validation process for the Tourism Recommender System 194
13.2 Example of records containing some user profiles (above) and a subset of the *T-Agent* ranking together with the corresponding user's feedback (below) 195
13.3 Distances of the *T-Agent* over the *S-cases* (left) and the corresponding frequencies (right). 197
13.4 Experimentation scheme 198
13.5 Distance frequencies for *T-Agent*, *T2-Agent* and *T3-Agent*. 201
13.6 Distance frequencies for the *T-Agent* family. 203
13.7 Distance frequencies for the *T2-Agent* family. 204
13.8 Total averages for the different agents. 206
13.9 Tukey's confidence intervals for the differences between *T-Agent* and other relevant agents ($\mu_{agent} - \mu_{T-Agent}$). 207
13.10 Tukey's confidence intervals for the differences between *T2-Agent* and other relevant agents ($\mu_{agent} - \mu_{T2-Agent}$). 208

List of Tables

9.1 Syntax of Ambient calculus 119
9.2 Syntax of Multi-context calculus (MCC) 123

11.1 RGC card for roles 160

13.1 Result of the *N-cases* in the validation process 196
13.2 Distance frequencies for *T-Agent*, *T2-Agent* and *T3-Agent*. 201
13.3 Distance average results for the *T-Agent* and *T2-Agent* families. . . . 205
13.4 Descriptive measures of the distances obtained over *S-cases*, using different agents. 206
13.5 ANOVA for the distances between the different agents' results and the S-cases 207

Part I

Introductory Concepts

The beginning is the most important part of the work.

Plato

Chapter 1

Introduction

Computer applications play an increasingly important role in everyday life. They are becoming more tightly connected with each other, forming large networks and interacting with humans through user-interfaces. Much of these systems are too complex to be completely characterized and precisely described; hence, these applications are hard to solve using centralized computing technology. Moreover, several of these systems are inherently distributed in the sense that the data and information to be processed is both geographically and temporarily distributed, or are structured into clusters whose access and use requires sophisticated capabilities [152].

We are confronted then with a new view of computing: computation as interaction, as an activity that is inherently social, and leading to new ways of conceiving, designing and developing computational systems. Agent based systems stand as a promising way to understand, manage and use these distributed, large-scale, dynamic, open and heterogeneous computing, information and social systems [85, 102]. Besides, multiagent systems offer a natural way of understanding and characterizing intelligent systems. Intelligence and interaction are deeply coupled and these systems enable us to model this insight. Several researchers argue that intelligent behaviour is not disembodied, but is a product of the interaction an agent maintains with its environment. Under this conception multiagent systems stand as a new approach to Artificial Intelligence [133].

With the spread of multiagent systems the number of projects and researchers involved in related fields has risen. Notably, there are some important coordination actions for agent based computing, as for instance AgentLink [1], an European network of researchers and developers with a common interest in agent technology. There are several websites providing information resources on intelligent agents, examples of these are Agentcities [2] and Agentland.[3]

The agent research community holds that there are some application domains where agent technologies will play a crucial role in the near future, including: am-

[1] http://www.agentlink.org

[2] http://grusma2.etse.urv.es/AgCitES/

[3] http://www.agentland.com

bient intelligence, grid computing, electronic business, the semantic web, bioinformatics, monitoring and control, resource management, space, military missions and manufacturing. The impact of agent technologies in application domains such as these will occur firstly as a design metaphor of complex distributed computational systems; secondly, as a source of technologies for such computing systems, and thirdly, as models of complex real-world systems [86, 102].

Considering agents as a design metaphor, they provide software designers and developers with a way of structuring an application around autonomous, communicating components, and lead to the construction of software tools and infrastructure to support this design. In order to support this view of systems development, particular tools and techniques need to be introduced. For example, methodologies that guide the analysis and design processes are required, agent architectures are needed for the design of individual software components, tools and abstractions are required to enable developers to deal with the complexity of implemented systems, and supporting infrastructure must be integrated.

In order to achieve the full potential of agent approaches and technologies there are a number of broad technological challenges for the near future. In the Agent Technology Roadmaps of the AgentLink network [86, 102], Luck et al. recommend that research and development resources should be focused along several key directions. Some of them are the following:

1. Creating tools, techniques and methodologies to support agent systems developers.

2. Automating the specification, development and management of agent systems.

3. Integrating components and features. Many different theories, architectures, technologies and infrastructures are required to specify, design, implement and manage agent based systems.

4. Establishing appropriate linkage with other branches of computer science and with other disciplines.

The work reported in this Thesis can be placed within these mentioned directions.

1.1 Motivations

This Thesis work undertakes an extension of the BDI agent architecture in order to incorporate the representation of uncertainty in beliefs, desires —thus allowing the expression of graded positive and negative desires, and graded intentions. Some years ago, I worked in research projects related to knowledge-based systems, studying how different approaches to approximate reasoning could be applied to them, and turning these systems more flexible and useful for real applications [103, 100]. Now, in the framework of multiagents systems, i.e. in a distributed and complex

platform of autonomous, proactive, reactive and social agents; I asked myself how the ideas underlying approximate reasoning could be extended and applied to these distributed systems.

Following this motivation, we found an interesting paper by Parsons and Giorgini [116] where a first approach to a graded BDI model was presented. It included only the representation of uncertainty in the beliefs, and left the general graded model as an open research problem. This "open door" to future work encouraged us to take this research direction. There are other contributions which treat agents that reason under uncertainty in dynamic and complex environments, but most of them deal with partial aspects of graded attitudes in intentional agents [97, 124, 142].

In this Thesis a graded BDI agent model is proposed, this model allows us to define concrete agents capable of dealing with uncertain environments (i.e. graded Beliefs) and with graded mental proactive attitudes (i.e. Desires and Intentions). Besides proposing an agent model, we consider it important to define its operational semantics to describe how a valid agent model is interpreted as sequences of computational steps. Notably, process calculi have been used to cope with formal aspects of multi-agent systems [130, 148] and we wondered if the same approach could be used to give this agent model computational meaning.

On the other hand, software engineering methodologies for developing agent based systems have become an important necessity. Even though there are valuable approaches in this field (e.g. [118, 157]), few of them emphasize the internal design of agents and consider a particular architecture. Furthermore, the actual engineering of graded BDI agents in a multiagent scenario was another relevant motivation for our work.

In the following Section, we pinpoint the contributions of this Thesis to these different fields.

1.2 Contributions

We consider that making the BDI architecture more flexible will allow us to design and develop agents capable of improved performance in uncertain and dynamic environments, serving other agents (human or not) that may have a set of graded motivations. In this research line, the central contribution of this work is the proposal of a graded BDI agent model (g-BDI), specifying an architecture capable of representing and reasoning with graded mental attitudes.

We consider this work to be an important contribution to the agent architectures field, because of the relevance of the BDI architecture and because some of our ideas may be adapted to other agent architectures. Moreover, we dealt with the operational semantics of the agent model as a first step towards a g-BDI agent interpreter and we also developed a methodology to engineer multiagent systems composed by g-BDI agents. In summary, the Thesis contributions are situated on the following diverse fields:

1. **Agent architectures:** *a general graded BDI agent model is proposed.*

   In this model, the agent graded attitudes have an explicit and suitable representation. Belief degrees represent the extent to which the agent believes a formula to be true. Degrees of positive or negative desire allow the agent to set different levels of preference or rejection respectively. Intention degrees also give a preference measure but, in this case, modelling the cost/benefit trade off of achieving an agent's goal. Then, agents having different kinds of behaviour can be modelled on the basis of the representation and interaction of their graded beliefs, desires and intentions.

   The specification of the g-BDI agent model is based on Multi-context systems (MCS). These systems were introduced by Giunchiglia et al. [68] to allow different formal (logic) components to be defined and interrelated, and Parsons et al. in [115] firstly used them to formalize BDI agents. The MCS specification of agents has several advantages pointed out by Sabater et al. in [136] both from a software engineering and a logical point of view.

   In order to represent and reason about graded notions of belief, desire and intention, in the g-BDI model we followed the approach developed by Godo et al. [72] where uncertainty reasoning is dealt with by defining suitable modal theories over suitable many-valued logics. This formalization permits us to deal with the different mental attitudes within the same well-founded logical framework.

   An illustration of the development of the g-BDI agent model and its related works is shown in Figure 1.1. The evolution of the g-BDI agent model, can be seen in [29],[30] and [31].

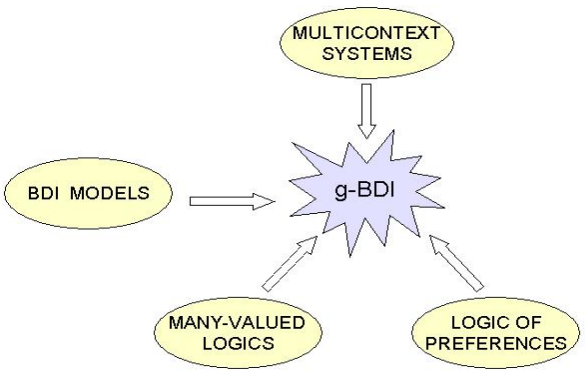

Figure 1.1: Related work to the g-BDI agent model development

2. **Knowledge representation and reasoning:** *a logical framework with a sound and complete axiomatics for representing beliefs, desires and intentions is presented.*

   Looking for suitable logical systems for representing and reasoning about beliefs, desires and intentions in the g-BDI agent model is a knowledge representation problem. The question of how to deal with uncertain beliefs has been widely studied in the AI community and several approaches to approximate reasoning have been proposed (as for instance see [79]). The problem of preference representation (i.e. desires and intentions) has been also approached in some works (e.g. [96], [11]). Considering the desire representation in our agent model, we based our work on the bipolar model due to Benferhat et al. [12] and we extend the state of the art by giving a sound and complete axiomatics and defining different logical schemas to represent some additional constraints over preferences. In addition, we present a logical system for intentions and we show that the framework is expressive enough to describe how desires (either positive or negative), together with other information, can lead agents to intentions. Recent work in this direction was presented in [37].

3. **Process calculi:** *a Multi-context calculus (MCC) to define operational semantics for multi-context systems is developed and we use it for giving semantics to the g-BDI agent model.*

   In order to cope with the operational semantics aspects of the g-BDI agent model, we first defined a Multi-context calculus (MCC) for Multi-context systems (MCS) execution. The calculus proposed is based on Ambient calculus [28] and includes some elements of the Lightweight Coordination Calculus (LCC) [148]. We expect that MCC will be able to specify different kinds of MCSs. Particularly, we have shown how graded BDI agents can be mapped into this calculus. Through MCC we give this agent model computational meaning and in this way, we move one step closer to the development of an interpreter of the g-BDI agents. Figure 1.2 illustrates the path leading to the operational semantics of the agent model. Although process calculi have been used in the past to model multiagent systems [130, 148], we have considered that the modular structure that MCS provides to the architecture of an agent would permit a similar treatment of single agents as well. Preliminary results on the language for the execution of g-BDI agents can be seen in [36].

4. **Agent based software engineering:** *a methodology for engineering agent based systems composed by agents designed as g-BDI agents, is presented.*

   We propose a software engineering process to develop graded BDI agents in a multiagent scenario. The aim of the proposed methodology is to guide the design of a multiagent system starting from a real world problem. This process is illustrated in Figure 1.3. The methodology presented has been built by adapting and extending previous approaches [88, 118, 159] in order

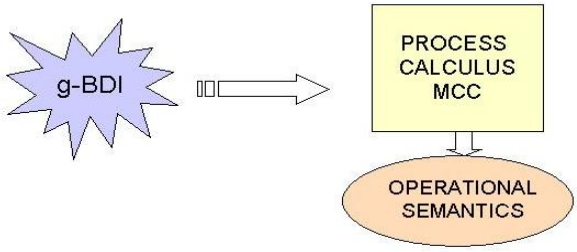

Figure 1.2: Giving Operational Semantics to the g-BDI agent model

to engineer agents with a more complex internal architecture. Furthermore, our work was inspired in some sense by the design process used in [141] where the social aspects of design are considered, and the system design phase is clearly separated from the agent design phase. Preliminary results on the methodology proposed can be seen in [34]. The design and implementation of a case study in the tourism domain is developed to show how the proposed methodology works.

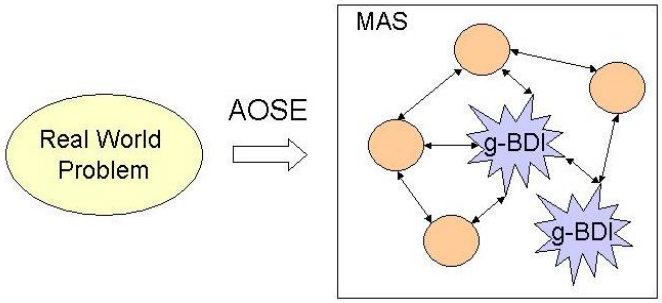

Figure 1.3: Software Engineering for the g-BDI agents

Through the design and implementation of a Tourism recommender system, where one of its principal agents is modelled as a g-BDI agent, we show in the first place that the agent model is useful to design and implement concrete agents in real world applications. The Tourism recommender design and implementation was presented in [32, 35].

Finally, using the case study we have made some experiments concerning the flexibility and performance of the g-BDI agent model. The experiments demostrate that this agent model is useful to develop concrete agents showing varied and rich behaviours. We also show that the results obtained by these particular recommender agents using graded attitudes improve those achieved by agents using non-graded

attitudes. The validation and experimentation of the g-BDI Model using the case study are exposed in [38].

1.3 Structure

This Thesis is structured in four main Parts. Part I, is about Introductory Concepts and apart from the current Introduction in Chapter 2, we present related work to our research, such as: agent theories and architectures, with special attention to the BDI model; the multi-context systems and their approach to agent specification and engineering; and some logics of preference. Then, in Chapter 3 we review some of the logical background which is fundamental for our work, i.e. dynamic logic to reason about the agent actions and the transformation they produce, and some many-valued logic as Godel logic, Rational Pavelka logic and Rational Lukasiewicz logic, to reason about fuzzy modal formulae in the different contexts.

In Part II, the Graded BDI agent model is presented and then, in the consecutive Chapters 4 to 7 the general framework, its fundamental components as the different Contexts (mental and functional) and the Bridge Rules are formalized. Later on, in order to give an idea of how this model works, we show an example of a travel assistant agent. Then, the extension of the basic model that includes social and dynamic aspects is considered in Chapter 8. In Chapter 9 the operational semantics of this agent model are given.

Next, in Part III the software engineering aspects are addressed. In Chapter 10 we present the characteristics of the Tourism domain where our case study is situated. Then, in Chapter 11 a methodology to engineer g-BDI agents in a multiagent system is developed and a case study is designed using the proposed methodology. At the end of this Part, in Chapter 12, a prototype implementation of the recommender tourism system is described.

Part IV is dedicated to present our experimental work in Chapter 13 and finally, in Chapter 14 the most relevant contributions related to our Thesis work are discussed and we present some lines of future work.

Is there anyone so wise as to learn by the experience of others?

Voltaire

Chapter 2

Related Work

2.1 Introduction

In this Chapter we introduce different lines of work relevant to the agent model proposed in this Thesis. Some fundamental theories that constitute the building blocks of our agent architecture, are presented. Firstly, the principal agent architectures and the theories supported them are revised, particularly we focus on the family of BDI agents, where our proposal is situated. In second place, the backgrounds of multi-context systems and their use on agent specification, are shown. Then, we revise some logics of preferences to represent and reason about the agent positive and negative desires. Finally, some approaches to graded attitudes in intentional agent architectures are discussed.

Besides, two important lines of related work are presented in next Chapters 9 and 11. These fields are not fundamental for the agent model definition, but are vital to give the agent model semantics and to state a methodology for its engineering. Then, in Chapter 9 we present some Process Calculus used for giving operational semantics to different systems, and particularly to formalize coordination characteristics in multiagent systems. Later on, in Chapter 11 some approaches to Agent based Software Engineering are presented to cope with the methodological aspects of agent based systems and particularly, of those composed by BDI agents.

2.2 Agent Theories and Architectures

In order to give multiagent systems a formal support, several researchers have proposed diverse theories and architectures for agents. Agent theories are essentially specifications of agents behaviour expressed as properties that agents should satisfy. A formal representation of the properties helps the designer to reason about the expected behaviour of the system.

Agent architectures can be thought of as software engineering models of agents and represent a middle point between specification and implementation. They iden-

tify the main functions that ultimately determine the agent's behaviour and define the interdependencies that exist among them. A relevant review of the work done on agent theories and architectures is due to Wooldridge and Jennings in [154].

Agent theories based on an *intentional stance* are among the most common ones. These are based on a folk psychology by which human behaviour is predicted and explained through the attribution of attitudes. For example, when explaining human activity, it is often useful and common to make statements such as the following:

Jorge took his coat because he *believed* it was going to be cold.

Peter worked hard because he *wanted* to save money.

In these examples, Jorge's and Peter's behaviours can be explained in terms of their attitudes, such as believing and wanting. The philosopher Dennet has coined the term *Intentional system* to describe entities "whose behaviour can be predicted by the method of attributing certain mentalistic attitudes such as belief, desires and rational acumen" [48]. Dennet also identifies different grades of intentional systems: a first-order intentional system has believes and desires (etc.) but no beliefs and desires about beliefs and desires. A second-order intentional system is more sophisticated; it has beliefs and desires (and possibly other intentional states) about beliefs and desires (and other intentional states), both those of others and its own.

When the underlying system process is well known and understood, there is no reason to take an intentional stance, but this is not the case in many applications. The intentional notions are abstraction tools, which provide with a convenient and familiar way of describing, explaining, and predicting the behaviour of complex systems. Considering that an agent is a system that is conveniently described by the intentional stance, it is worth to weigh up which attitudes are appropriate for representing agents. The two most important categories are *information attitudes* —knowledge and belief— and *pro-attitudes* —desire, intention, obligation, commitment, choice, among others.

Information attitudes are related to the knowledge that the agent has about the world, whereas *pro-attitudes* are those that in some way guide the agent actions. The attitudes of both categories are closely related and much of the work in agent theory is concerned with clearing up the relationships between them. Although there is no total agreement on which combination of attitudes is the most appropriate to characterize an agent, it seems reasonable that an agent must be represented in terms of at least one information attitude and one pro-attitude.

There are various formalisms that focused on just one aspect of agency (i.e., beliefs, desires, intentions, etc.) but it is expected that a realistic agent theory will be represented in a logical framework that defines how the attributes of agency are related; how an agent cognitive state changes over time; how the environment affects the agent beliefs; and how the agent information and pro-attitudes lead it to perform actions [154].

Considering now the area of agent architectures, this field represent the move from specification to implementation. We need to address such questions as: How are we to construct computer systems that satisfy the properties specified by a particularly agent theory? How the agent can be decomposed into the construction of a set of component modules and how these modules should interact? What software/hardware structures are appropriate?

Maes defines an agent architecture as

> "A particular methodology for building agents. It specifies how...the agent can be decomposed into the construction of a set of component modules and how these modules should interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current mental state of the agent determine the actions...and the future mental state of the agent." (Maes [101], p115).

Making specific commitments about the internal structure and operation of agents, we have a distinct class of agents. There exists different proposal for the classification of agent architectures. Following the classification defined by Wooldridge in [152, 158], we consider four classes of architectures for intelligent agents:

1. *Logic based architectures(deductive agents)*
2. *Reactive architectures (reactive agents)*
3. *Layered architectures (hybrid agents)*
4. *Practical reasoning architectures (BDI agents)*

In the rest of this Section we outline the main characteristics of each kind of architecture and in the following Section 2.2.5 we present the BDI model in more detail.

2.2.1 Logic-Based Architectures - *Deductive agents*

A classical approach to build agents follows the traditional way for building artificial intelligent systems. This paradigm suggests that the intelligent behaviour can be generated in a system by giving it a symbolic representation of its environment and its desires, and allowing it to syntactically manipulate this representation.

A deductive or deliberative agent is one that contains an explicitly represented, symbolic model of the world, in which the decisions are made through logical reasoning, based on pattern matching and symbolic manipulation. In most cases, these symbolic representations are logical formulae and the syntactic manipulation corresponds to logical deduction or theorem proving. The idea of deliberative agents as theorem provers is attractive and a number of more-or-less "pure" logical approaches

to agent programming have been developed. However, there are still several problems associated with this approach to agency to be solved (many of them come from the symbolic approach to AI):

The transduction problem: how to translate the real world into an accurate, adequate symbolic description, in time for that description to be useful.

The representation/reasoning problem: how to symbolically represent knowledge about complex and dynamic real-world entities and processes, and how to get agents reason with this knowledge in time.

Calculative rationality: the assumption that the world will not change in any significant way while the agent is making decisions. This is not acceptable in dynamic environments that change faster.

Computational complexity: the complexity of theorem proving makes it questionable whether agents using this deduction mechanism can operate effectively in time-constrained environments.

2.2.2 Reactive architectures - *Reactive agents*

The problems with symbolic or logical approaches to build agents led some researchers to proclaim that a whole new proposal was required. They began to investigate different alternatives to the symbolic AI paradigm. Although it is difficult to characterize these different approaches, they agree in a number of points:

- the rejection of symbolic AI (as a representation-reasoning mechanism).
- intelligence and rational behaviour are not disembodied (they are a product of the interaction the agent maintains with its environment).
- intelligent behaviour emerges from the interaction of various simpler behaviours.

Alternative approaches to agency are sometimes referred to as *behavioral* — since these agents develop and combine individual behaviors, *situated* —since they are actually situated in some environment, rather than being disembodied from it, and finally *reactive*, the name used to represent this class of agents, because these systems are often perceived as simply reacting to the environment, without reasoning about it.

One of the best-known alternative architecture is the *subsumption architecture*, developed by Brooks [23], one of the most influential critics of the symbolic approach to agency in the last years. This architecture is also called behavior-based architecture and some authors, as Wooldridge in [152, 158], classified it as a reactive one. There are two defining characteristics of the subsumption architecture. The first one is, that an agent's decision making is realized through a set of *task accomplishing behaviours*. Each behaviour may be though it of as an individual action function, it

takes perceptual input and maps it to an action, neither does it include any complex symbolic representation nor reasoning. Each of these behaviour modules is intended to achieve some particular task. In Brooks' implementation these modules are finite state machines. The second important characteristic is, that many behaviours can fire simultaneously. Hence, there must be a control mechanism to choose among the different actions selected. Brooks proposed to organize the modules into a *subsumption hierarchy*, with the behaviours arranged into *layers* —the lower the layer is, the higher is its priority. Another characteristic of the subsumption systems implementation is that there is assumed to be a quite tight coupling between perception and action, and there is no attempt to transform the input data to symbolic representations.

One of the principal advantages of the reactive approaches over the logic-based ones is that the complexity is tractable. Other advantages of these approaches such as Brooks' subsumption architecture are: simplicity, economy and robustness against failure. However, there are some fundamental unsolved problems related to the reactive architectures that are remarked in [158]:

- If reactive agents do not employ models of their environment, then they must have sufficient information available to determine an acceptable action
- How reactive agents would take into account global information as these agents make decision based just on local information.
- How to incorporate learning from experience is not addressed.
- It is difficult to engineer this kind of agents to fulfill specific tasks and there is no principled methodology for building reactive agents. In purely reactive systems the overall behaviour emerges from the interaction of component behaviours when the agent is placed in its environment. Sometimes the relationships between individual behaviours, environment and the overall behaviour is not understood.
- It is hard to build agents that contains many layers. Effective agents can be generated with small —less than ten— numbers of behaviours.

2.2.3 Layered Architectures - *Hybrid agents*

Many researches have argued that neither a complete reactive nor deliberative approach is suitable for building agents. Given the requirement that an agent must be capable of reactive and proactive behaviour, an interesting approach involves creating separate subsystems to deal with these different kinds of behaviours. A class of architectures in which the defined subsystems are arranged into hierarchy and interacting *layers*, implements this idea.

In this approach, an agent will be defined in terms of two or more layers, to deal with the reactive and pro-active behaviours, respectively. The agent control

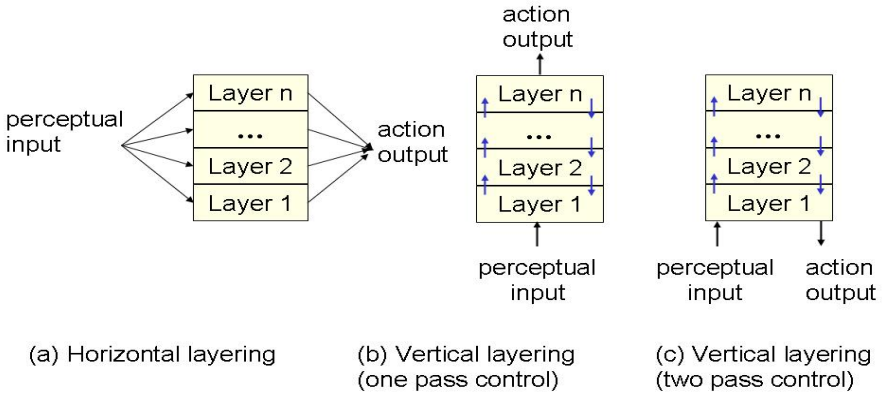

Figure 2.1: Information and control flows in layered agents architecture (Source: [111], p263).

subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction. An important problem in such architectures is to determine what kind of control framework is needed, in order to manage the interactions between the various layers. Two basic types of control flow can be identified within layered architectures, as it is shown in Figure 2.1 and are described in [158]:

- *Horizontal layering:* each layer is directly connected to the sensory input and action output, acting like an agent and producing action proposals (Figure 2.1 (a)).
- *Vertical layering.* Sensory input and action output are each dealt with by at most one layer. In this case there are two approaches:
  - one-pass architecture: control flows sequentially through each layer, until the final layer (Figure 2.1 (b)).
  - two-pass architecture: control flows up the architecture (the first pass) and then, control flows back down (Figure 2.1 (c)).

The great advantage of the horizontally layered architecture is its conceptual simplicity. One layer can be implemented for each behaviour the agent needs to exhibit. The different layers may generate competitive actions suggestions, sometimes inconsistent. In order to ensure the system consistence, it generally includes a mediator function. This function decides which layer has control on the agent at any time, the design of this function is difficult. This problem is in part solved in the

vertically layered architecture where the complexity of interactions between layers is reduced. However, the vertical layering has a disadvantage: in this architecture the control must pass between each different layer and a failure in any one layer will affect the whole agent performance. Examples of the layered architectures are Ferguson's TouringMachines —horizontally layered architecture— [59], and Muller's InteRRaP —two-pass vertically layered— [112].

2.2.4 Practical Reasoning Architectures - *BDI agents*

Practical reasoning is a particular model of decision making. This model is inspired in the process that seems to take place when we decide what to do next —the process of deciding which action to perform in order to reach our goals. The philosopher Michael Bratman defines this process as:

> "Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/care about and what the agent believes." (Bratman, [22], pp.17)

Practical reasoning involves two important processes: *deliberation* —deciding what goals or desires we want to achieve, and *means-ends reasoning* —how we are going to achieve them. After generating these set of alternative goals, the agent must choose among them, and commit to some. These goals committed to achieve are the agent's intentions. In practical reasoning process, intentions play a crucial role lead to actions.

Specifically, Bratman argues that rational agents will tend to focus their practical reasoning on the intentions they have already adopted, and will tend to avoid the consideration of options that conflict with them. Some of the most relevant points of his work are known as Bratman's claim, and we can summarize their characteristics as follows [158]:

- *Intentions drive means-ends reasoning:* If an agent has an intention, then it will attempt to achieve it, which involves deciding how to achieve it. If one way fails to achieve an intention, then it will attempt others.

- *Intentions persist:* The agent will not give up on its intention without a good reason —it believes it cannot achieve them or that the reason for the intention is no longer present.

- *Intentions constrain future deliberation:* The agent will not consider options that are inconsistent with its current intentions, and

- *Intentions influence beliefs upon which future practical reasoning is based:* The agent can plan for the future on the assumption that she will achieve her current intentions.

Bratman's theory of intention [20, 22] is an important contribution and his work points out that intentions play an fundamental role in practical reasoning. In turn, the agent intentions interact with and sometimes depends on, the agent beliefs and desires. However, in the design of practical reasoning agents satisfactorily capturing these interactions and achieving a good balance among the different items presented above, turn out to be considerably difficult.

Some of the philosophical aspects of a rational agency were well formalized by Cohen and Levesque [41, 42]. They developed a *Logic of Rational Agency* where they provided one of the first logical formalization of intentions and the notion of commitment using just two basic attitudes: beliefs and goals (i.e., desires). Other attitudes, as intentions, were defined in terms of these. This theory of intention and commitment was applied as for example, to the formalization of communicative actions among agents [42].

In particular, intentions are modeled as a kind of commitment (i.e., persistent goal) and are defined in terms of temporal sequences of an agent beliefs and goals. The mechanism an agent uses to determine when and how to drop intentions is known as commitment strategy. Based on different kinds of commitments specific agents are proposed [123]:

- *Blindly commited agent* is a fanatically committed agent that will continue to mantain an intention until she believe the intention has actually been achieved.
- *Single-minded commited agent* will continue to maintain an intention until either it is believed to be achieve or it is believed to be unachievable.
- *Open-minded commited agent* is an agent with a *relativized commitment* to her intentions is similar to the other but, may also drop her intentions when some specified conditions are believed to hold.

A key problem in the design of practical reasoning agents implementing this kind of commitments (in particular the single-minded and open-minded ones) is that they must revise their intentions. Specially it seems clear that the agent should at times drop some intentions (because the reasons mentioned above) It follows that, from time to time it is worth an agent stopping to reconsider its intention. But reconsideration has a cost (i.e. in time and resources). This problem is known in the literature as *intention reconsideration* (e.g. see [142, 143]) and treat the problem of balancing the agent pro-active (goal-directed) and reactive (event driven) behaviour, in relation to the environment dynamism.

While the Cohen and Levesque's formalization treats intentions as being reducible to beliefs and desires, Bratman [20] argues that intentions play a significant and distinct role in practical reasoning. He also shows how the agent current beliefs, desires and intentions, constitute a background for future deliberations. Systems and formalisms that give primary importance to intentions represent an important

class of the BDI architectures.

The Belief-Desire-Intention (BDI) architecture was originated in the work of the Rational Agency Project at Stanford Research Institute in the mid-1980s. This agent model is based on the theory of human practical reasoning [20, 22] mention above. Within the "Agent Theory, Architectures and Languages" (ATAL) community [66], the BDI model has come to be possibly the best known and studied model of practical reasoning agents. There are several reasons for its success, but perhaps the most compelling are that the BDI model combines a respectable philosophical model [20, 22], a number of implementations (e.g. [20], [61] and [18]), several successful applications [61], and finally, and a well-founded logical semantics [126, 139].

In the Agent Community, the term BDI model is used in different ways, including a family of agent models and architectures. In a wide sense, they are models of practical reasoning that employ the folk-psychology concepts of belief, desire and intention, perhaps among other attitudes. In a narrow sense, there are particular BDI models that embody Bratman's claim, as for example, the IRMA specific architecture (for the "Intelligent Resource-Bounded Machine Architecture" described in [20]). Also, there are particular BDI models that suitable specified Procedural Reasoning System (PRS), as for instance [61, 62]. In this Thesis we adopt the broader position, calling BDI models, to those models of practical reasoning that explicitly represent the agent mental attitudes (i.e. belief, desire and intentions).

The basic components of a BDI architecture are data structures representing the beliefs, desires and intentions of the agent, and functions that represent its deliberation —deciding what intentions to have, and means ends reasoning —deciding how to do them. We present a general formalization of this agent model, describing its components and their relations, following Wooldridge in [158].

Let Bel be the set of all possible beliefs, Des be the set of all possible desires, Int be the set of all possible intentions, P is the current percept and the set A of the actions the agent can execute. The state of a BDI agent at any given point of time is a triple (B, D, I), where $B \subseteq Bel$, $D \subseteq Des$ and $I \subseteq Int$. The process of practical reasoning in a BDI agent may be summarized in the schema shown in Figure 2.2. This Figure illustrates the main components in a BDI agent that are described as follows:

- a *set of current beliefs (B)*, representing the information the agent has about its environment,

- a *belief revision function (brf)*, which takes a perceptual input and the agent current beliefs, and determines a new set of beliefs:
  $brf : \wp(Bel) \times P \rightarrow \wp(Bel)$,

- *an option generation function (options)*, which determines the options available to the agent (its desires D), on the basis of the current beliefs and its

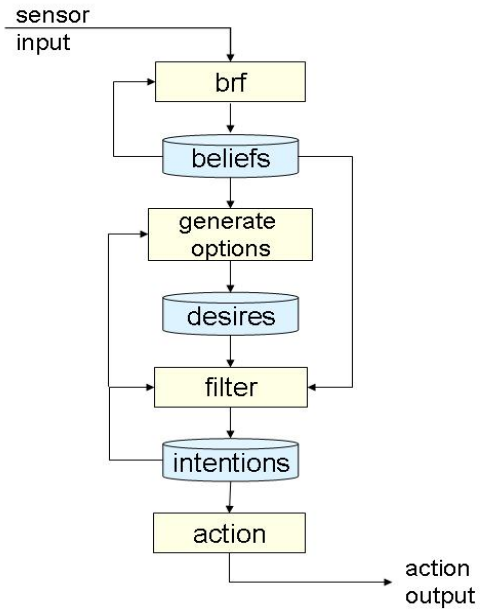

Figure 2.2: Schematic diagram of a generic belief-desire-intention architecture (Source: [152], p58).

current intentions:
$options : \wp(Bel) \times \wp(Int) \rightarrow \wp(Des)$,

- a *set of current options (D)*, representing possible course of actions available to the agent,

- a *filter function (filter)*, which represents the agent deliberation process in which the agent determines its intentions, based on its currents beliefs, desires and intentions:
$filter : \wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int)$

- a set of *current intentions (I)*, representing those states of affairs that it has committed to try to bring about.

- an *action selection function (execute)*, which determines an action to perform on the basis of current intentions.
$execute : \wp(Int) \rightarrow A$

The resulting process is the agent decision function $action : P \rightarrow A$. This function maps the input perception into an action that the agent will try to execute and is defined in terms of the data structures and functions previously presented. A simple version of this function is defined by the following pseudocode:

```
function ACTION (p:P):A
      B:= brf(B,p)
      D:= options(D,I)
      I:= filter(B,D,I)
      return execute(I)
end function ACTION
```

A more complete version of this practical reasoning loop, including intention reconsideration, could be seen in ([158], p76).

Procedural Reasoning Systems

The procedural Reasoning Systems (PRS), originally developed by Georgeff and Lansky [61] was one of the first agent architecture that explicitly embody the BDI model.

The PRS is a BDI architecture because it contains explicitly represented data structures loosely corresponding to these mental attitudes. An illustration of the PRS architecture is given in Figure 2.3 PRS is a goal-directed and reactive planning

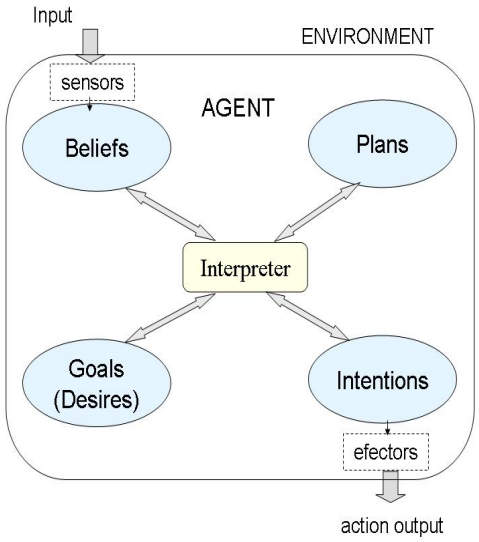

Figure 2.3: The procedural Reasoning System

system. The goal-directed behaviour allows to reason about and to perform complex tasks, while reactiveness allows handling real-time behaviour in dynamic environment In the PRS an agent does no planning instead, it is equipped with a library of precomputed plans which are used to to perform means-ends reasoning. The process of selecting between different possible plans is a deliberation process and includes the use of *meta-level* plans which are able to modify an agent's intention structure

at runtime, in order to change the focus of the agent's practical reasoning. Beliefs in the PRS are represented as PROLOG-like facts (i.e. atoms of a first order logic).

Since the first PRS system [61], it has been re-implemented several times afterwards, as for example the dMARS system [46], implementations in C^{++} are UM-PRS and Open-PRS and a Java version is called Jam system [81], Besides, two relevant implementations currently very used are Jack [65] and Jason [18]. Furthermore, it has been applied in several of the most significant multiagent applications so far built including an air-traffic control system called OASIS, a simulation system for the Royal Australian air force called SWARMM, the now-famous fault diagnosis system for the space shuttle, as well as factory process control systems and business process management called SPOC, overviews of these systems are described in [62]).

The BDI model is also interesting because a great deal of effort has been dedicated to formalize it. In particular, Anand Rao and Michael Georgeff have developed a range of BDI logics, which they use to axiomatized properties of BDI agents. In the following Subsection, we outline the more important features of its logical framework.

2.2.5 Rao and Georgeff's BDI model

One of the well-known intentional system formal approach that follows Bratman's claim, was proposed by Rao and Georgeff [123, 125]. This BDI model is built upon the theory of Intentional systems.

This model is based on the explicit representation of the agent *beliefs* (B), *desires* (D) and *intentions* (I), using a logical framework based on the possible world semantics. Firstly, Rao and Georgeff in [123] present the logic formalism in terms of the agent *belief*, *goals* and *intentions*. Then, they revised the concept of the agent goal and it was replaced in [125] by the concept of desire, as it is still used nowadays. In this section we used [123] as reference of the BDI logic and then, the term of goal is used instead of desire.

In the design of rational agents the role played by attitudes such as beliefs (B), desires (D) and intentions (I) has been well recognized and analyzed by philosophical and AI researchers. The beliefs are needed to represent the state of the world, the desires, to set the state of affairs the agent wants to achieve and the intentions, the worlds the agent has chosen and is committed to achieve. In this formalism intentions are represented as a fundamental attitude like beliefs and desires.

Rao and Georgeff used to model the world as a temporal structure with a branching time future and a single past, called a *time tree*. The branches in a time tree can be viewed as representing the choices available to an agent at each moment of time. A particular time point in a particular world is called *situation*. *Events* transform the state at one time point into another state at a subsequent time point. *Primitive events* (actions) are those events that the agent can execute directly, and uniquely determine the next time point in a time tree. *Non-primitive events* (plans) map to non-adjacent time points. The agent may attempt to execute some event, but

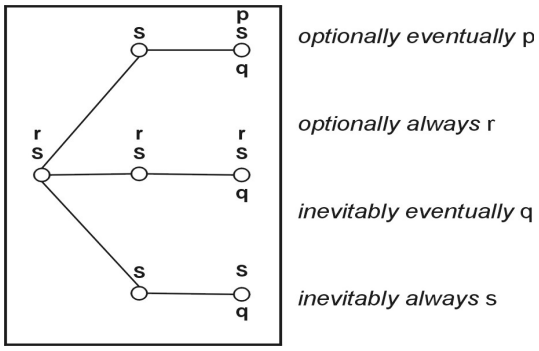

Figure 2.4: Belief-accessible world (Source: [123]).

may fail to do so (i.e., successful execution of events or their failure). They use a formalism similar to Computation Tree Logic, CTL [53] to describe these structures. A distinction is made between *state formulae* —evaluated at a specified time point in a time tree, and *path formulae* —over a specified path in a time tree.

The modal operators *optional* and *inevitable* are used to operate on path formulae. A path formula ψ is said to be *optional* if, at a particular time point in a time tree, ψ is true in at least one path emanating from that point; it is *inevitable* if ψ is true in all paths emanating from that point. They also used the standard temporal operators O (next), $\Diamond$ (eventually), $\Box$ (always) and U (until), in order to operate over *state* and *path* formulae.

These modalities can be combined in various ways to describe the options available to the agent. For example, the structure illustrated in Figure 2.4 could be used to represent the following statements [123]:

- *it is optional that John will eventually visit London (denoted by p);*
- *it is optional that Mary will always live in Australia (r);*
- *it is inevitable that the world will eventually come to an end (q) and*
- *it is inevitable that one plus one will always be two (s).*

Belief is modelled in the conventional way, in each situation they associate a set of belief-accessible worlds and each belief-accessible world is a time tree. Multiple belief-accessible worlds result from the agent lack of knowledge about the state of the world. This approach take into account the uncertainty in the agent beliefs allowing a set of possibles world, but in this approach they do not use a belief measure (e.g., a probability measure, possibility measure, etc) to establish an order over the set of worlds (i.e., expressing which of these are the most believable ones). Within each of these worlds, the branching future represents the choice (options) still available to the agent in selecting which actions to perform. In similar way, for each situation they associate a set of goal-accessible worlds to represent the goals of the agent. They use goals as a set of chosen consistent desires.

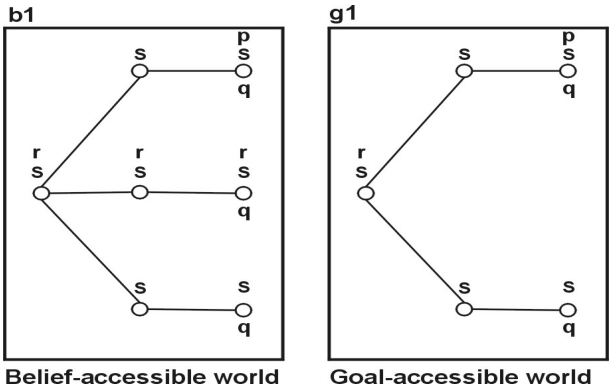

Figure 2.5: Compatibility between Belief and Goal-accessible world (Source: [123]).

There are three well-established sets of attitudes relationships for the BDI agents that has been identified in [123]. These three types of agents incorporates different kind of relations between the attitudes (i.e. belief, desire and intention -accessible worlds) called realisms, and they are define as follows:

- *Strong realism:* the set of intentions is a subset of desires which in turn is a subset of the beliefs. That is, if an agent does not belief something is possible to become true, it will neither desire nor intend it.

- *Realism:* the set of beliefs is a subset of desires which in turn is a subset of the set of intentions. That is, if an agent beliefs something is possible, it both desires and intends it.

- *Weak realism:* agents do not desires properties if the negation of those properties are believed, do not intend properties if the negation of those properties are desired, and do not intend properties if the negation of those properties are believed.

The formalization of these realism in a multi-context specification of BDI agents can be seen in next Subsection 2.3.2).

In this review, we adopt the notion of *strong realism*. This sets up a relation between the belief- and goal-accessible worlds: it is required that the agent believes she can optionally achieve her goals. This kind of belief-goal compatibility is illustrated in Figure 2.5. Intentions are similarly represented by a set of intention-accessible worlds. These worlds are ones that the agent has committed to attempt to realize. The intention-accessible worlds of the agent must be compatible with her goal-accessible worlds. In the case of a *strong realism* agent, she can only intend some course of action if it is one of her goals.

It thus remains to formalize this semantics presented informally a the introduction to the BDI logical model.

BDI logic

BDI logic is a branching-time temporal logic (CTL) extended in two ways. First, they consider a first-order variant of the logic, and second, it is extended to a possible world framework by introducing modalities for the believes, desires and intentions.

Then, its language includes the temporal operators U, $\Box$, $\Diamond$, O, *optional* and *inevitable*, also modalities BEL, $GOAL$ and $INTEND$ to represent the mental attitudes beliefs, desires and intentions, respectively. We present this formalization as in

There are two types of formulae in the logic: *state* formulae and *path* formulae. A *state* formula is defined as follows:

- any first-order formula is a *state* formula,
- if φ and ψ are *state* formulae and x is variable, then $\neg\varphi$, $\varphi \vee \psi$, and $(\exists x)\psi(x)$ are *state* formulae,
- if e is an *event* type then succeeds(e), fails(e), does(e), succeeded(e), failed(e), and done(e) are *state* formulae,
- if ψ is *state* formula then $BEL(\psi)$, $GOAL(\psi)$ and $INTEND(\psi)$ are *state* formulae and
- if ψ is a *path* formula, then $optional(\psi)$ is a *state* formula.

A *path* formula can be defined as follows:

- any *state* formula is also a *path* formula and
- if Φ and Ψ are *path* formulae, then $\neg\Phi$, $\Phi \vee \Psi$, Φ U Ψ, $\Diamond\Psi$ and OΨ are *path* formulae.

There are some abbreviations used in the language for representing some formulae, namely:

- $\Box(\psi)$ for $\neg\Diamond(\neg\psi)$.
- $inevitable(\phi)$ for $\neg optional(\neg\phi)$
- $done(e)$ for $succeeded(e) \vee failed(e)$
- $succeeds(e)$ for $inevitableO(succeeded(e))$
- $fails(e)$ for $inevitableO(failed(e))$
- $does(e)$ for $inevitableO(done(e))$

As for example, *optionally*$\Diamond p$, *optionally*$\Box r$, *inevitably*$\Diamond q$ and *inevitably*$\Box s$ are *state* formulae that are true at the root state (time t_0) of the world shown in Figure 2.4.

BDI Semantics

The formalization of this semantics is presented by Rao and Georgeff in [123]. First, they provide the semantics of the different formulae, secondly of the events and finally, the possible world semantics of beliefs, goals, and intentions. In the following we briefly outline this schema:

An interpretation M is defined as $M = (W, E, T, \prec, U, B, G, I, \Phi)$, where:

- W is a set of possible worlds,
- E is a set of primitive event types,
- T is a set of time points,
- $\prec$ a binary relation on time points,
- U is the universe of discourse,
- Φ is a mapping of first-order entities to elements in U for any given world and time point, and
- $B, G, I \subseteq W \times T \times W$ are accessible relations for BEL, GOAL and INTEND, respectively.

  Notation: R refers to any one of these relations (B,G,I) and $R^{w,t}$ to denote the set of worlds R-accessible from world w at time t:

  $$R^{w,t} = \{w' : R(w, t, w')\} \ for \ R = B, G, I$$

  As for example, we show in Figure 2.6 the relation B^{w,t_2} including the worlds w' and w''.

Each world $w \in W$, called a time tree, is a tuple (T_w, A_w, S_w, F_w), where:

- $T_w \subseteq T$ is a set of time points in the world w,
- A_w is the restriction of $\prec$ to T_w,
- $S_w : T_w \times T_w \rightarrow E$ map adjacent time points to (successful) events in E and
- $F_w : T_w \times T_w \rightarrow E$ map adjacent time points to (failing) events in E.

  The domains of S_w and F_w are disjoint.

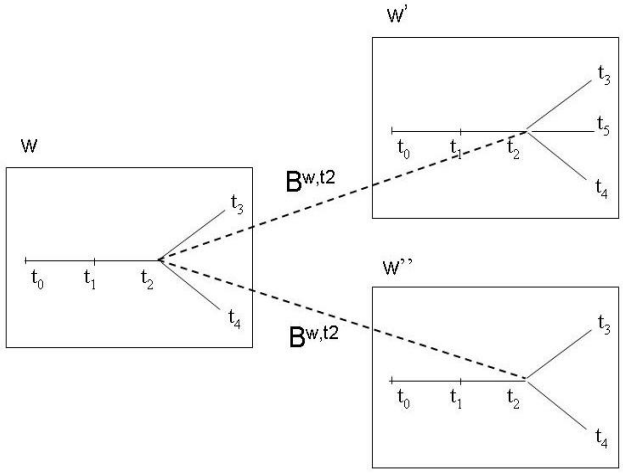

Figure 2.6: Belief-accessible world relation

A *fullpath* in world w is an infinite sequence of time points $(t_0, t_1, \ldots)$ such that $(t_i, t_{i+1}) \in A_w$ y fullpath $(t_0, t_1, \ldots)$ in world w is denoted as: $(w_{t_0}, w_{t_1}, \ldots)$.
Considering an interpretation M and a variable assignment v, the semantics of the *state* formulae are defined as following:

- $M, v, w_t \models q(y1, ..., yn) \Leftrightarrow (v(y_1), ..., v(y_n)) \in \Phi[q, w, t]$ where $q(y1, ..., yn)$ is a predicate formula.
- $M, v, w_t \models \neg\phi \Leftrightarrow M, v, w_t \not\models \phi$
- $M, v, w_t \models \phi \vee \psi \Leftrightarrow M, v, w_t \models \phi$ or $M, v, w_t \models \psi$
- $M, v, w_t \models (\exists x)\phi \Leftrightarrow M, v\,[d/x]\,, w_t \models \phi$ for some $d \in U$
- $M, v, w_t \models optional(\phi) \Leftrightarrow$ exists a full path $(w_{t_0}, w_{t_1}, ...)$ such that $M, v, (w_{t_0}, w_{t_1}, ...) \models \phi$

Semantics of state formulae pertaining to events:

- $M, v, w_{t1} \models succeeded(e) \Leftrightarrow$ exists t_0 s.t. $S_w(t_0, t_1) = e$
- $M, v, w_{t1} \models failed(e) \Leftrightarrow$ exists t_0 s.t. $F_w(t_0, t_1) = e$

Semantics of Belief, Goals and Intentions:

The possible world semantics of beliefs, considers each world to be a collection of propositions and models belief by a belief-accessibility relation B linking these worlds. In this BDI model, each possible world is a time tree and denotes the optional courses of events that an agent can choose in a particular world. The belief

relation maps a possible world at a time point to other possible worlds. An agent has a belief ϕ, denoted $BEL(\phi)$, at time point t if and only if ϕ is true in all the belief-accessible worlds of the agent at time t. The semantics of the modal operator $GOAL$ is given in terms of a goal-accessible relation G which is similar to that of the B relation. The goal-accessibility relation specifies situations that the agent desires to be in. Intentions can be seen as future paths that the agent chooses to follow. The intention-accessibility relation will be used to map the agent's current situation to all its intention-accessible worlds. Formally, this semantics is defined as follows:

- $M, v, w_t \models BEL(\phi) \Leftrightarrow \forall w' \in B_t^w,\ M, v, w'_t \models \phi$
- $M, v, w_t \models GOAL(\phi) \Leftrightarrow \forall w' \in G_t^w,\ M, v, w'_t \models \phi$
- $M, v, w_t \models INTEND(\phi) \Leftrightarrow \forall w' \in I_t^w,\ M, v, w'_t \models \phi$

The semantics of path formulae:

- $M, v, (w_{t_0}, w_{t_1}, ...) \models \phi \Leftrightarrow M, v, w_{t_0} \models \phi$ (ϕ state formula)
- $M, v, (w_{t_0}, w_{t_1}, ...) \models O\phi \Leftrightarrow M, v, (w_{t_1}, w_{t_2}, ...) \models \phi$
- $M, v, (w_{t_0}, w_{t_1}, ...) \models \Diamond\phi \Leftrightarrow M, v, (w_{t_k}, ...) \models \phi$ for some $k \geq 0$
- $M, v, (w_{t_0}, w_{t_1}, ...) \models \phi$ U $\psi \Leftrightarrow$
  - exists $k \geq 0$ s.t. $M, v, (w_{t_k}, ...) \models \psi$ and $\forall 0 \leq j \leq k$ $M, v, (w_{t_j}, ...) \models \phi$, or
  - $\forall j \geq 0,\ M, v, (w_{t_j}, ...) \models \phi$

Axiomatization

The basic axiomatization for beliefs is the classic weak-S5 modal system or KD45. For goals and intentions the K and D axioms are adopted to make them closed under implication and satisfy the consistence condition. The rule of necessitation is also needed for beliefs, goals and intentions (i.e., the agent believes, has as goal, and intends all the valid formulae). As happens in most possible world formalisms, this logic suffers from the logical omniscience problem i.e., the agent believes, desires and intends all the logical consequences of its beliefs, desires and intentions. Then, the axiom schema is the following:

- $BEL(\phi \rightarrow \psi) \rightarrow (BEL\phi \rightarrow BEL\psi)$ (axiom K)
- $BEL\phi \rightarrow \neg BEL(\neg\phi)$ (consistency - axiom D)
- $BEL\phi \rightarrow BEL(BEL\phi)$ (positive introspection - axiom 4)
- $\neg BEL\phi \rightarrow BEL(\neg BEL\phi)$ (negative introspection - axiom 5)

- $GOAL(\phi \rightarrow \psi) \rightarrow (GOAL\phi \rightarrow GOAL\psi)$
- $GOAL\phi \rightarrow \neg GOAL(\neg\phi)$
- $INTEND(\phi \rightarrow \psi) \rightarrow (INTEND\phi \rightarrow INTEND\psi)$
- $INTEND\phi \rightarrow \neg INTEND(\neg\phi)$
- Necessitation rule for beliefs, goals and intentions (from ϕ derive $BEL\phi$, $GOAL\phi$ and $INTEND\phi$)

In addition, Rao and Georgeff in [123] presented a set of axioms (A11 and A12) in order to set the interrelations among an agent's beliefs, goals and intentions. They also added an axiom that from leads intentions to actions (A13), and two axioms (A14-A15) to establish that the agent believes what it is intending its goals. This group of axioms is:

- (A11) $GOAL(\alpha) \rightarrow BEL(\alpha)$ (belief-goal compatibility)
- (A12) $INTEND(\alpha) \rightarrow GOAL(\alpha)$ (goal-intention compatibility)
- (A13) $INTEND(does(e)) \rightarrow does(e)$ (intention leading to action)
- (A14) $INTEND(\phi) \rightarrow BEL(INTEND(\phi))$
- (A15) $GOAL(\phi) \rightarrow BEL(GOAL(\phi))$
- (A16) $INTEND(\phi) \rightarrow GOAL(INTEND(\phi))$
- (A17) $done(e) \rightarrow BEL(done(e))$ (awareness of primitive events)
- (A18) $INTEND(\phi) \rightarrow inevitable\Diamond(\neg INTEND(\phi))$
  (no infinite deferral)

This set of eight axioms A11-A18 together with the standard axioms for BDI logics (KD45 for BEL and K-D for GOAL and INTEND) constitute the basic I-system. Furthermore, Rao and Georgeff analyzed in [123] the relation between current and future intentions —commitment strategy— in a process of intention maintenance and revision. They described three different commitment strategies: *blind*, *single minded* and *open minded*. A *blindly committed* agent is one who maintains its intentions until the agent actually believes that they have been achieved. A *single-minded committed*, is an agent which maintains its intentions as long as its believes that they are still options. Finally, an *open-minded* agent is one who maintains its intentions as long as these intentions are still its goals. In order to obtain one of these different behaviours in an agent, the corresponding axioms must be added to the basic I-system:

- for a Blind agent:
  $INTEND(inevitable\Diamond\phi) \rightarrow$
  $inevitable(INTEND(inevitable\Diamond\phi)$ U $BEL(\phi))$.

- for a single-minded agent:
  $INTEND(inevitable\Diamond\phi) \rightarrow$
  $inevitable(INTEND(inevitable\Diamond\phi))$U $(BEL(\phi) \vee \neg BEL(optional\Diamond\phi))$.

- for an open-minded agent:
  $INTEND(inevitable\Diamond\phi) \rightarrow$
  $inevitable(INTEND(inevitable\Diamond\phi))U(BEL(\phi) \vee \neg GOAL(optional\Diamond\phi))$.

2.2.6 Advantages of BDI models

Several factors have contributed to the importance of the BDI model. This architecture is one of the best models of practical reasoning that is based on well understood logical foundations. The BDI model has proved to have the essential components to cope with complex, real world applications. These real systems are usually placed in a dynamic and uncertain environment, having a local view of the world and are resource bounded. These constrains have certain fundamental implications for the design of the underlying computational architecture, and the Belief, Desire and Intention components seem to be an essential part of such systems.

The BDI model is also interesting because a great deal of effort has been done in its formalization. In particular, Rao and Georgeff have developed a range of BDI logics. They set out different axiomatics over the basic logic, called I-system, to define BDI agents having different properties (e.g., diverse commitment strategies).

But the importance of the BDI models is not limited to the theoretical field. In the last years there have been different developments of particulars BDI architectures. One of the specific BDI agent architecture is IRMA [21], this architecture has been evaluated in an experimental scenario known as the Tileword. However, the best-known implementation of the BDI model is the Procedural Reasoning System (PRS) system developed by Georgeff and Lansky [61] and re-implemented several times afterwards (e.g. [81, 65, 18]. This agent architecture has proved to be the most durable agent architecture developed to date. It has been applied in several of the most significant multiagent applications built up to now, some of them are described in [62]. In addition, PRS-like systems has been used in a number of large-scale applications, as for example, a system for space shuttle diagnosis and a system for telecommunications network management [84]. The BDI architecture has evolved over time and diverse factors, including the above mentioned ones, have contributed to the importance of this model.

Because of the recognized relevance of the BDI model we decided to use this agent architecture as the basis for this PhD research work. In the following subsection we

introduce one interesting approach to specify complex systems and particularly, agent architectures.

2.3 Multi-context Systems

The notion of context has been studied in many research areas an in particularly in Artificial Intelligence. Contexts are view as an important approach to represent certain kinds of reasoning. On the one hand, contexts are a tool to formalize the locality of reasoning. While on the other hand, contexts are introduced as a means of solving the problem of generality. Coherently with these two points of view, Giunchiglia et al. in [70, 71] introduced the notion of *multi-context system* (MCS for short). These systems have also been called *multi-language systems* in [71], in order to emphasized that they may include multiple languages.

There are two main intuitions underlying the use of contexts, called principles in [68]:

- *Locality principle*: reasoning uses only part of what is potentially available (e.g.,what is known, the available inference procedures). The part being used while reasoning, is what we call context (of reasoning);
- *Compatibility principle*: there is a compatibility among the kinds of reasoning performed in the different contexts.

These two principles are formalized by the semantics called *Local Model Semantics*, which is described in [68]. In this paper the authors also showed how this novel semantics is captured by the MCS. They also validate this semantics by formalizing two important forms of contextual reasoning: reasoning with viewpoints and reasoning about belief.

One of the advantages of MCS in order to help in the design of complex logical systems is that this framework allows for the independent definition of formal components, and their interrelations.

MCS have been used in diverse applications as for example in the integration of heterogeneous knowledge and data bases, in the formalization of reasoning about beliefs (more generally, propositional attitudes) [68]. Particularly, have been used to model different aspects of agents and multiagent systems [40, 68] and as an approach to engineering multiagent systems [119, 136].

2.3.1 Formalization of multi-context systems

We present an introduction to the formal aspects of MCS systems, where contexts are formalized proof-theoretically. A more complete description is given in [71]. The MCS specification contains three basic components: units or contexts, logics, and bridge rules, which channel the propagation of consequences among theories.

Following this, a MCS is defined as a group of interconnected units:
$\langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, where:

- for each $i \in I$, $C_i = \langle L_i, A_i, \Delta_i \rangle$ is an axiomatic formal system where L_i, A_i and Δ_i are the language, axioms, and inference rules respectively. They define the logic for the context C_i and its basic behaviour is constrained by the axioms.

- Δ_{br} is a set of bridge rules, they are rules of inference which relate formulae in different units. Each *bridge rule* can be understood as a rule of inference with premises and conclusions in different contexts, for instance:

$$\frac{C_1 : \psi, C_2 : \varphi}{C_3 : \theta}$$

  means that if formula ψ is deduced in context C_1 and formula φ is deduced in context C_2 then formula θ is added to context C_3.

When a theory $T_i \subseteq L_i$ is associated with each unit, the specification of a particular multi-context system is complete.

These components were first identify in the context of theorem provers for modal logic in [71], and in [114] a full detail of these components can be found.

A MCS system is essentially a set of logical theories, plus a set of inference rules which allow for the propagation of consequences among theories.

The deduction machinery Δ in these systems is then based on two kinds of inference rules, internal rules Δ_i inside each unit, and bridge rules Δ_{br} outside, i.e.,

$$\Delta = \bigcup_{i \in I} \Delta_i \cup \Delta_{br}$$

Internal rules allow to draw consequences within a theory, while bridge rules allow to embed results from a theory into another. The set of formulae that a given context may contain depends on its initial theory, axioms, inference rules that allow inner deductions; and bridge rules. The formulae introduced by a bridge rule depend on the formulae contexts appearing in the premise of the bridge rule.

A MCS formalizes the principle of locality in the sense that each context has it suitable language L_i, the proper set of axioms A_i which provides the foundations of reasoning, the theory $T_i \subset L_i$ gives the true formulae for each context, and finally, the inference engine Δ_i, captures different deduction capabilities for each unit. Through bridge rules the principle of compatibility is represented as these rules allow contexts to mutually influence themselves. Bridge rules change the theory of one context by the derivation of formulae in other contexts.

In the following subsection we show how this kind of systems, that have been used in diverse applications, can be used in agent specification.

2.3.2 Multi-context agents

Multiagent systems are complex systems than can be well modeled by MCSs, as they permit to represent the locality of its architectural components and to neatly describe the interaction among them. This approach has been used by Parsons et al. [119] and Sabater et al. [136] to specify several agent architectures and particularly to model some classes of BDI agents [115]. Using the multi-context approach, an agent architecture consist of the four basic types of components of MCS (i.e., contexts, logics, theories and bridge rules).

Contexts represent the various components of the architecture. They contain the agent's problem solving knowledge, and this knowledge is encoded in the specific theory that the unit encapsulates. In general, the nature of the contexts will vary between architectures.

For example, a BDI agent may have units which represent intentional notions —theories of beliefs, desires and intentions— (as in [115]), whereas an architecture based on a functional separation of concerns may have units which encode theories of cooperation, situation assessment and plan execution (as in [138]). In either case, each context has a suitable logic associated with it.

In any architecture represented, bridge rules set the components interaction. These rules provide the mechanism by which information is transferred among units. The bridge rules continuously examine the theories of the contexts that appear in their premises looking for new sets of formulae that match them. This means that all the components of the architecture are always ready to react to any change (external or internal) and that there are no central elements of control.

The multi-context approach was used to specified negotiating agents in an example of two Home Improvement Agents, described in [115]. An extended model of multi-context agent was presented in [138] to engineer the ReGreT system.

Multi-context BDI agents.

The BDI architecture was described in Subsection 2.2.5. This agent model involves the explicit representation of the agent beliefs, desires and intentions. In a logical framework this means to include different modalities for the different attitudes and the chosen axiomatic according to the behaviour of each attitude. Modeling different intentional notions by means of several modalities (e.g., B, D and I) can be very complex if an unified logical framework is used (e.g., the BDI logic see Subsection 2.2.5) or if one must manage the interchange of formulae among different logics.

Using multi-context systems to build BDI agents allow as to represent the different mental attitudes by different contexts. This is advantageous with respect to other approaches as pointed out in [136] and exemplified in [115]. The MCS approach enables to use different logics in a way that keeps the logics neatly separated. This either makes it possible to increase the representational power of BDI agents —compared with those which use a single logic, or to simplify agents conceptually —compared with those which use several logics in one global framework.

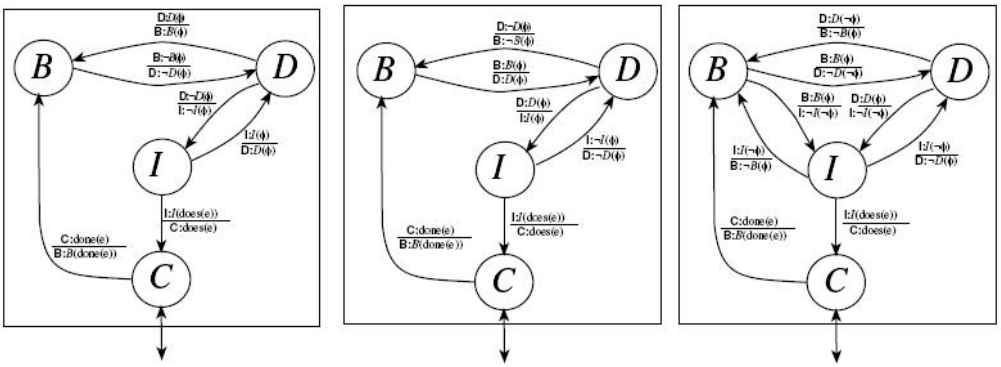

Figure 2.7: Different types of BDI agent. From left to right, the relations between modalities correspond to strong realism, realism and weak realism. (Source [115], p272).

Thus, in a MCS approach, we need at least different contexts to represent the three basic attitudes i.e., one for beliefs (B), for desires (D) and intentions (I). The belief context of a BDI agent may have a logic of belief associated with it, the desire context may have a logic of preferences associated to it, and similarly for the intention unit. The logic related with each unit provides the language in which the information in that context is encoded.

We have presented in Subsection 2.2.5 three well-established sets of attitudes relationships for the BDI agents, called realisms. A multi-context version of these types of agent (i.e strong realistic, realistic and weak realistic) are formalized in [115] and are illustrated in Figure 2.7.

As to show how this approach may be implemented, we present some insights of the multi-context BDI agent model developed by Parsons et al. in [115]. The specification corresponds to a strong realistic BDI agent and its main components can be seen in Figure 2.7 (left) and are defined as follows:

Contexts: There are four contexts within a multi-context BDI agent. The units for the beliefs (B), desires (D) and for the intentions (I); and a communication unit (C).

Logics: For each of these four contexts a proper logic is defined:
- *B, D and I context:* each one uses first-order logic with a special predicates B, D and I, which are used to denote respectively the beliefs, desires and intentions of the agent. The chosen axioms are the classics for predicate logics. To capture the behaviour of the modalities, in the B context the KD45 axioms are included, and in the logics for D and I, the axioms K and D are used.
- *Communication context:* Uses classical first-order logic with the usual axioms.

The rules of inference for each unit are the usual ones (MP, MT, generalization

and particularization)

Theories: For each context, these logical formulae represent the domain information that each unit posses, and depend on the specific agent we are defining (in a generic BDI agent there are no specific theories included).

Bridge rules: The bridge rules are exactly those illustrated in Figure 2.7 for a strong realism BDI agent, formally:

$$\begin{aligned} I : I(\alpha) &\Rightarrow D : D(\lceil \alpha \rceil) \\ D : \neg D(\alpha) &\Rightarrow I : \neg I(\lceil \alpha \rceil) \\ D : D(\alpha) &\Rightarrow B : B(\lceil \alpha \rceil) \\ B : \neg B(\alpha) &\Rightarrow D : \neg D(\lceil \alpha \rceil) \\ C : done(e) &\Rightarrow B : B(\lceil done(e) \rceil) \\ I : I(\lceil does(e) \rceil) &\Rightarrow C : does(e) \end{aligned}$$

The first four rules are directly derived from the model proposed by Rao and Georgeff and ensure compatibility between what the agent believes, desires and intends. The last two bridge rules specify the interactions that the communication context has with the other units.

Concrete agents may be specified as extensions of this generic specification of a BDI agent. The complete specification of two home improvement BDI agents is presented in [115].

For BDI agents, the multi-context approach make it possible to model each agent attitude in an appropriate local way, and the corresponding interactions between attitudes are neatly represented through the bridge rules. It also allows the incorporation of other attitudes to the agent model by just adding the corresponding contexts and the necessary bridge rules, relating the new attitude with the rest.

2.3.3 Advantages of the multi-context specification of agents

Multi-context approaches to engineering multiagent systems has several advantages, some of them are pointed out by Sabater et al. in [136]. From a software engineering perspective, firstly, MCSs support the development of modular architectures. Each architectural component, be it a functional component or a data structure component, can be represented as a separate context. The interrelations between the components can then be made explicit by writing bridge rules to link the contexts. This ability to directly support component decomposition and interaction offers a path from the high level specification of the architecture to its detailed design. Secondly, MCSs are ideally suited to support reuse —both of designs and implementations— since these systems encapsulate architectural components and provide specifications for the interrelationships.

From the logical modeling perspective, there are several advantages of adopting a multi-context approach. In first place, separating the logical description of an agent into a set of contexts, each which its proper logic, we effectively get a form of many-sorted logic (all the formulae in one context are a single sort). This brings to the system the advantages of scalability and efficiency. The second advantage comes from the same issue. This approach makes it possible to build agents which use several different logics in a way that keeps the logics neatly separated (all the formulae in one logic are gathered in one context).

The remaining two advantages from the logical perspective apply to those logical agents which reason about their mental attitudes and those of other agents. The first is that multi-context systems make it possible to build agents which reason in a way which conforms to the use of modal logics like KD45 (the standard modal logic for handling belief) while working within the computationally simpler framework of standard predicate logic [71]. The final advantage is related to this. Agents which reason about beliefs are often confronted with the problem of modeling the beliefs of other agents and multi-context systems provide a neat solution to this problem [40, 68].

Combining the software engineering and the logical modeling perspectives, it can be seen that the multi-context approach offers a clear path from specification to implementation. Indeed one advantage of the MCS logical approach to agency modeling is that allows for rather affordable computational implementation. For instance, a portion of the framework described in [115] has been implemented using a prolog multi-threaded architecture [69].

2.4 Logics of preference

Preferences guide human decision making from early childhood (e.g. "which ice cream flavour do you prefer?") to complex professional and organizational decisions (e.g."which investment funds to choose"). Preferences are essential for making intelligent choices in complex situations, for mastering large sets of alternatives, and for coordinating a multitude of decisions. Explicit preference models allow an agent to reason about its own and the other agent's behaviour and to analyze and revise this behaviour. For these reasons, preference models have been necessary in many fields of Artificial Intelligence such as multiagent systems, combinatorial auctions, diagnosis, design, configuration, planning, among others. In addition to this, preference modelling and aggregation is central to decision theory, social choice and game theory. AI tasks often need new forms of preference handling beyond classic utility-based models. Recent work on preference handling in AI has consequently elaborated many new preference representation formalisms, as for example logical preference representations and generalized forms of utility functions. AI has also innovated reasoning about preferences and problem-solving algorithms based on preferences. This is an important issue when we have to represent the user's desires

in information systems (e.g. recommender systems), or to reason about desires and solve eventually inconsistent goals as e.g., in multiagent systems. Logical frameworks contribute to this problem, allowing for systematic study and classification of desires by making underlying assumptions explicit. Recently, several logics for desires and goals have been proposed as can be seen in [9, 19, 49, 95, 96, 146].

Notably, Lang et al. in [96] propose a logic of desires with a utilitarian semantics and they study non-monotonic reasoning about desires and preferences. Their work is about representation and reasoning on conditional desires $D(a \mid b)$ and is based on the idea that desires can be understood in terms of so-called utility losses and gains (i.e. *loss of utility* resulting from its violation $\neg a \wedge b$ and/or a *gain of utility* from its fulfillment $a \wedge b$). According to these utility components (losses and gains) they distinguish three types of desires (gain, loss and mixed) then, to sum up these utilities additive functions are used. Furthermore, they propose different procedures to induce, from a set of initial desires, the preference relation of the agent on a set of worlds. Namely, in this approach a set of desires induces a set of utility functions by adding up the utility losses and gains of the individual desires, and these distinguished utility functions induce a (qualitative) partial preference ordering on worlds. They also consider how to introduce domain knowledge expressing which worlds are feasible in this ordering process. In contrast with Boutiller [19] they also differentiate between factual background knowledge, telling us which worlds are physically impossible, and contingent knowledge, expressing which of the physical possible worlds can be the actual state of affairs. Then, the agent should attempt to the best feasible world by performing a suitable action. This work is not focussed on action theories but they use an action model inspired in [19] to represent feasible worlds. Despite this is a valuable contribution to desire representation, we found in this approach some limitations in relation to the desire representation we want for our agent model. First, it does not include an explicit logical representation of the agent rejections (negative preferences). Second, from qualitative expressions about desires, they give as result a (qualitative) preference order over worlds. Then, this proposal does not treat with different numerical levels of desires, useful for deciding, besides other factors, the agent intentions. Finally, the authors recognize that the role of expressing desirability is only a partial account of the use of desires in the agent decision process toward intentions and present a more complex schema (involving beliefs and actions), as future work.

Later on, the authors extend in [97] their qualitative logical approach to desire representation, introducing the notion of hidden uncertainty of desires. The semantics of this logic is defined by means of two ordering relations representing preference and normality as in Boutilier's logic QDT [19]. Desires are formalized to support a realistic interaction between the concepts of preference and plausibility (or normality), both represented by a pre-order relation over the sets of possible worlds. Their idea is to express desires with a suitable order modality $<_{nd}$ where $A <_{nd} B$ means that, taking into account normality (expressed by another order relation $\geq_N$),

A is less desirable than B. This work considers that an ordinal-like uncertainty is present in the notion of plausibility, whose corresponding pre-order may be defined by the proximity of the current world to the set of most plausible (or normal) worlds.

Besides these approaches to preference representation, we observe that real-life problems present positive and negative preferences. Relevant works on bipolar preference representation focuss on the fact that preferences over solutions or choices are often expressed in two forms: positive and negative aspirations. On the one hand, an agent may express what he considers unacceptable (to some degree) and on the other hand, it may express what it considers desirable or satisfactory (to some level). The first kind of preferences are called *negative preferences* and correspond to constraints that should be respected, while the second type are called *positive preferences* and correspond to desirable states of the world for the agent.

For instance, assume that we want to take a week of holidays and we are looking for a tourist destination in the country. We may provide the tourism agent with two kinds of preferences. First, we specify the satisfactory slots, with different desire levels (e.g. we strongly prefer mountains, moderately prefer small cities and we weakly like to make rafting), these are positive preferences. Second, we describe unacceptable conditions, that are refused in different degrees (e.g. we do not want to travel more than 1000 km), these are negative preferences or rejections. Then, the agent is expected to find the best desirable solutions (e.g. tourist destinations) among the feasible ones (i.e. not satisfying any negative desire).

This bipolar representation is supported by recent studies in cognitive psychology showing that the distinction between positive and negative preferences makes sense. They are processed separately in the brain, and are felt as different dimensions by people [26, 27]. Note that in general there is no symmetry between positive and negative preference in the sense that positive desires do not mirror what is not rejected. The idea of bipolar representation of preferences has been considered in different works as the ones oriented towards qualitative decision making based on ordinal rankings [51, 146]. Besides, Bistarelli et al. in [16] propose a generalization of the soft constraint formalism, which is able to model problems with one kind of preferences (constraints) allowing to handle positive preferences as well. Also, the use of bipolar information (prioritized desires and rejections) in an argument based decision framework can be seen in [122]. Particularly, Benferhat et al. [10, 11, 12] present a valuable approach, a bipolar possibilistic logic framework for modeling preferences. As this approach inspired or work on agent desire representation, we describe some of its fundamental aspects on the following Subsection.

2.4.1 Bipolar representation of preferences

This bipolar approach to preferences is supported by the work by Benferhat et al. [10] on modelling positive and negative information. They presented a framework based on possibility theory where this distinction can be made in a graded way. In

logical terms, the two types of information —positive and negative— can be encoded by two types of constraints expressed by necessity measures and other possibility functions. Particularly, they applied this model to the representation and fusion of preferences, and also they show how this bipolar information can be used to take optimal solutions. The description of the bipolar representation of preferences in the possibilistic logic framework can be seen in detail in [11, 12], we briefly outline the relevant features of their approach.

The syntactic specification of this bipolar representation of preferences is done introducing two different sets of constraints. These sets correspond to what the agent rejects and what are its goals or desires, respectively:

- $R = \{R(\phi_i) \geq \alpha_i,\ i = 1, ..., n\}$, where ϕ_i is a propositional formula, $\alpha_i \in [0, 1]$ and reflects the agent rejection strength of ϕ_i. The higher α_i is, the less acceptable are the solutions satisfying ϕ_i. $R(\phi_i) = 1$ means that the agent strongly rejects ϕ_i, and no solution where ϕ_i is true is tolerated by the agent. This rejections can be also encoded as $R = \{(\neg\phi_i, \alpha_i),\ i = 1, ..., n\}$ where $(\neg\phi_i, \alpha_i)$ represent the constraint $R(\phi_i) \geq \alpha_i$ and stands for "if a solution w satisfies ϕ_i then it is tolerated at most to a degree $1 - \alpha_i$".

It turns out that the set of rejections can be handled using the classical possibility and necessity measures.

- $G = \{G(\psi_j) \geq \beta_j,\ j = 1, ..., m\}$, where ψ_j is a propositional formula, $\beta_j \in [0, 1]$ and expresses the minimum level of satisfaction guaranteed to the agent, if ψ_j is true. Thus ψ_j is supposed to encode a desire or a wish, β_j expresses the minimal level of satisfaction which is guaranteed for a solution where ψ_j is true. The larger β_j is, the more satisfied is the agent if ψ_j is true. $G(\psi_j) = 1$ means that the agent is fully satisfied if ψ_j is true. G may be also denoted as $G = \{[\psi_j, \beta_j] : j = 1, ..., m\}$. Thus, $[\psi_j, \beta_j]$ encodes the information $G(\psi_j) \geq \beta_j$.

This kind of positive desires (goals) cannot be directly handled by the classical possibilistic logic machinery, as it is explained below.

Representing rejections in possibilistic logic.

Rejections can be represented, at semantical level, by a total pre-order on the set of possible outcomes (interpretations). This pre-order can be encoded in possibility theory using a possibility distribution over the set of worlds $\pi_R : W \rightarrow [0, 1]$. This possibility function π_R associated with a set of rejections $R = \{R(\phi_i) \geq \alpha_i, i = 1, ..., n\}$, is defined as:

$$\pi_R(w) = 1 - max\{\alpha_i : w \models \phi_i, R(\phi_i) \geq \alpha_i \in R\}, \quad with\ max\{\emptyset\} = 0$$

Clearly, this definition can be viewed in terms of a necessity measure replacing ϕ_i by $\neg\phi_i$ (if $R(\phi_i) = \alpha_i$ then $N(\neg\phi_i) \geq \alpha_i$)

Representing positive desires

The positive desires can also be described in terms of a possibility distribution: $\pi_G : W \rightarrow [0, 1]$, where $\pi_G(w) \geq \pi_G(w')$ means that w is more satisfactory for the agent than w'. The meaning of $\pi_G(w)$ is different from $\pi_R(w)$, the first evaluates to what degree w is satisfactory for the agent, while $\pi_R(w)$ evaluates to what extend w is acceptable.

The possibility degree π_G associated with a set of positive goals $G = \{[\psi_j, \beta_j], j = 1, ..., m\}$ is:

$$\pi_G(w) = max\{\beta_j : w \models \psi_j, [\psi_j, \beta_j] \in G\}, \quad with\ max\{\emptyset\} = 0$$

The addition of positive goals in G can only lead to the increase of the satisfaction level of w and this is dual to the behaviour of π_R which monotonically decreases with respect to the number of constraints in R.

The set of positive desires cannot be directly handled by standard possibilistic measures. Constraints like $G(\psi_j) \geq \beta_i$ are then represented using a function called *guaranteed possibilistic* function, denoted by Δ, first presented by Dubois and Prade in [50] and afterwards, used in [10] to represent bipolar information. The expression $\Delta(\psi) = b$ means that any interpretation where ψ is true, has a satisfaction degree at least equal to b, then:

$$\Delta(\psi) = min_{w \models \psi} \pi_G(w)$$

Hence, for the disjunction and conjunction Δ behaves as follows :

- $\Delta(\phi \vee \psi) = min(\Delta(\phi), \Delta(\psi))$, so Δ decreases with respect to disjunction. The semantic for disjunctions goes here in an opposite way than in classical logic. This means that $\phi \vee \psi$ are guaranteed to be possible —$\Delta(\phi \vee \psi) > 0$— (i.e., because they are observed, feasible, satisfactory, according to the problem) if and only if both ϕ and ψ are guaranteed to be possible $\Delta(\phi), \Delta(\psi) > 0$.

- $\Delta(\phi \wedge \psi) \geq max(\Delta(\phi), \Delta(\psi))$ since the minimum π_G over the worlds satisfying $\phi \wedge \psi$ may be greater than the minima over the worlds satisfying ϕ and ψ. Applying the maximal specifity principle on Δ, this inequality leads to the equality: $\Delta(\phi \wedge \psi) = max(\Delta(\phi), \Delta(\psi))$

Coherence relation between positive and negative preferences

Even if independently specified, negative and positive preferences must nevertheless be in agreement with each other. The author in [11, 12] proposed the following restriction between them "a solution cannot be at the same time unacceptable and desired by the same agent". Let (R, G) be an agent positive and negative preferences, intuitively if $R = \{R(\phi_i) \geq 1, i = 1, ..., n\}$ and $G = \{[\psi_j, 1], j = 1, ..., m\}$ (with maximal rejection or satisfaction degrees), then R and G are coherent if

$$\bigvee_{j=1,\ldots,m} \psi_j \vdash \bigwedge_{i=1,\ldots,n} \neg\phi_i$$

namely any solution satisfying at least one goal of G should not satisfy any formulae in R. More generally all solution that is satisfactory to a degree α should be at least feasible (tolerant) to a degree α.

From a semantic point of view if π_G and π_R are the two possibility distributions representing respectively the agent positive and negative preferences. Then, π_R and π_R are said to be coherent iff:

$$\pi_G(w) \leq \pi_R(w), \quad for\ all\ world\ w$$

Merging multiple agents preferences in a bipolar representation

Benferhat et al. in [11, 12] also treat the problem of merging multiagents preferences from a semantic and a syntactic point of view. The result of the merging process will also be a pair ($R_{\oplus_R}$, $G_{\oplus_G}$) where $R_{\oplus_R}$ is the result of merging negative preferences expressed by several agents, and $G_{\oplus_G}$ is the result of merging the agents positive preferences. These two merging steps are processed separately and generally use different operators (i.e. $\oplus_R$ and $\oplus_G$).

Merging agents' preferences can lead to conflicts. They discuss how to revise the set of positive desires when it is not coherent with the negative ones (as the negative preferences are considered strong constraints).

Finding the best solution according to bipolar preferences

The problem of computing the best solutions after merging negative and positive preferences separately can be viewed as an optimization problem involving the sets R, G and a set of integrity constraints F (representing domain knowledge). Different strategies are presented to compute the solutions that do not violate integrity constraints, avoid all the the negative preferences R and satisfy as many as possible agent's positive preferences G. Among the presented strategies we find *tolerate solutions* (satisfying R and F), *tolerate solutions satisfying at least one important positive preference* (e.g. positive preference having the highest degree in G) and *cardinality-based* selection mode (maximizing the number of respected rejections and maximizing the number of satisfied positive preferences).

In summary, the work by Benferhat et al. on bipolar representation of preferences propose a separate treatment of positive and negative information under the form of two sets of weighted logical formulae having different semantics. Both kinds of preferences are encoded in the framework of possibility theory. The bipolar representation of preferences allows to easily define different selection modes to find the

best solutions, according to the agent preferences and domain knowledge (coded as integrity constraints).

Finally, this approach inspired us to model, in our g-BDI architecture, the agent desires in a bipolar way (i.e. positive and negative) as it is shown in Chapter 6.

2.5 Graded Attitudes in Intentional Agent Architectures

If we want that agent technologies increase their role in complex and real applications, the complexity of the real-world environments where the agents interact, has to be considered. Most of the environments are not completely accessible, non-deterministic and dynamic. Moreover, the preferences or goals of agents (humans or not) interacting in the environment may be expressed with different levels of intensity. This means that there is uncertainty involved not only in the agent's model of the world, but even there are different degrees related to its pro-attitudes. In order to improve the agents performance, we consider essential to take into account this uncertainty and graded attitudes in the agent's theory, architecture and implementation.

Focussing on intentional agents and specifically on BDI model of agents, the agent architectures proposed so far mostly deal with two-valued information. We think that taking into consideration graded information (related to the different attitudes) could improve the agent's performance. Even in AI a lot of work have been done related to uncertain beliefs (e.g. see [79]) and to preference representation (see Section 2.4), in the context of intentional agent architectures, we found there are a few works that partially address this issue and emphasize the importance of graded models. There are some approaches considering graded information related to a particular attitude (i.e. belief, desire or intention), the relevant features of some of them are presented bellow.

In the BDI model developed by Rao and Georgeff, they explicitly acknowledge that an agent's model of the world is incomplete, by using a branching-time possible world logic to model the beliefs, goals (desires) and intentions. For each situation they associate a set of belief-, goal- and intention-accessible worlds; intuitively, those worlds that the agent believes to be possible, desires to bring about, and commits to achieve, respectively. Multiple possible world results from the agent lack of knowledge about the state of the world. Within each of these possible worlds, the branching future represents the choice of actions available to the agent. In a first proposal of the BDI model [123], they also considered the incompleteness of the agent's model of the world. However, they make no use of quantified information to assess how much a particular world is possible. Neither do they allow for desires and intentions to be quantified.

Afterwards, in [124] the authors extend the expressive power of BDI logic, by introducing subjective probabilities and subjective payoffs to model the process of

deliberation. Intuitively, an agent at each situation has a probability distribution on its belief-accessible worlds. The agent then chooses sub-worlds of these belief-accessible worlds that it considers are worth pursuing, and associates a payoff value to each path. Using a probability distribution on its belief-accessible worlds and the payoff value with each path in its goal-accessible worlds, the agent determines the best plan(s) of action for different scenarios. This process is called *Possible World (PW)* deliberation and is inspired by decision tree theory. The result of this process is a set of the most desirable sub-worlds of the goal-accessible worlds. These sub-worlds are the intention-accessible worlds that the agent commits to achieve. In this work the similarity between the PW-deliberation on the one hand, and the decision tree formalism on the other hand, is shown. We consider that this is an interesting approach, although it has some shortcomings. The first one is that they introduce the concepts of probability and payoff in the unified BDI logic framework thus, increasing its complexity. Second, the semantics of the payoff function over the path formulae is not clear. We think the payoff implicitly combines a kind of benefit of achieving some world with the cost of the path. But, as its meaning is not clear, we consider that it may be difficult to determine the function values, and sometimes may be unnatural. Besides, they don't use any measure degrees to represent the intentions in order to obtain an explicitly ordered set of possible intentions as the results of the deliberation process. And finally, the functions they use in the deliberation process are not neatly related to the BDI model.

Notably, Parsons and Giorgini [116] consider belief quantification by using Evidence Theory. In their proposal, an agent is allowed to express its opinion on the reliability of the agents it interacts with, and to revise its beliefs when they become inconsistent. The paper combines previous authors' works on the use of argumentation in BDI agents with other approaches to belief revision and update. The model presented is an extension of the multi-context specification of BDI agents developed in [115] to include degrees of belief. In order to introduce the degrees of belief they translate every proposition in the belief unit (which may contain nested modalities) into an argument with an empty set of grounds and with an associated degree of belief expressed as a mass assignment in Dempster-Shafer theory (i.e. $B(\Phi)$ becomes the argument: $B((\Phi) : \{\} : \alpha)$ where α is the belief degree). Any propositions deduced from the belief base set will then accumulate grounds. The belief revision process they used consists of redefining the degrees of credibility of propositions in the light of incoming information. Finally, the authors set out the importance of quantifying degrees in desires and intentions in a BDI agent model, but this is not covered by their work.

In the previous Section 2.4 we have revised different approaches to preference representation and reasoning that may be applied to desire representation in a BDI architecture. In the utilitarian logic of preference by Lang et al. [96] it is presented how a preference order over worlds can result from qualitative agent desires.

They present a general scheme of how this preference order in addition to agent knowledge (about feasible worlds, plans, etc.) can be used to decide the agent intentions. But this global agent decision procedure, related to action theory is left as future work. We remarked the importance of the bipolar representation of preferences due to Benferhat et al. [11]. In this bipolar approach the representation of weighted positive and negative preferences is formalized and we found this approach suitable to model the agent desires (both positive and negative). The authors also propose some alternative ways to find the best preferred solutions considering integrity constraints based on domain knowledge (feasible worlds). But before applying some of the ideas exposed in the different approaches to preference representation in an intentional agent model, there are many problems to be solved. For example, how to use uncertain domain knowledge and planning theory to find feasible worlds? How to take into account, beside the agent preferences, the cost of plans and also the possibility of plan failure, in deciding the agent intention? More importantly, how the agent can use intentions to derive the best action to follow?

After deciding the agent intention, a problem related to how long an agent must maintain her commitment to it is known as *intention reconsideration*. There has been a certain amount of work on the intention reconsideration problem, as for instance in [123] (see Subsection 2.2.4), where different commitment strategies were defined and in [156], where a formal perspective is presented. More recently, Parsons et al. in [117] addressed the intention reconsideration in environments which are both complex and dynamic. Other works deal with reasoning about intentions in uncertain domains, as the proposal of Schut et al [142]. They present an efficient intention reconsideration for BDI agents that interact in an uncertain environment in terms of dynamics, observability, and non-determinism. In this approach they consider that the internal state of an agent consist of beliefs and intentions: $s = \langle Bel, Int \rangle$. The agent beliefs are represented by a probability distribution $Bel : E \rightarrow [0, 1]$ where E is the set of environment states. The agent set of intentions Int is a set of environment variables. They assume that it is possible to assign values $V(i)$ and cost $C(i)$ to the outcomes of intentions and they define the *net value* , V_{net}, representing the net value of the outcome of an intention i: $V_{net} = V(i) - C(i), i \in Int$. They also express how good is a state defining a *worth* function: $W : S \rightarrow \mathbb{R}$, the value for each state s is based on the net value of the intentions of the state. From this state representation, they model the intention reconsideration by using the theory of Markov decision processes for planning in partially observable stochastic domains (POMDP). They used a POMDP approach because the optimality of the policy in this framework is based on the same three environment characteristics considered for the intention reconsideration strategy, namely: dynamism, determinism and observability.

Notice that all the above mentioned proposals related to graded beliefs, weighted preferences and intention reconsideration, model partial aspects of the uncertainty related to mental notions involved in an intentional agent architecture.

On the other hand, argumentation is a promising approach for reasoning with inconsistent information, based on the construction and the comparison of arguments [52, 56]. These approaches make possible to assess the reasons (i.e. arguments) why a fact holds, and to combine and compare these arguments in order to reach a conclusion. Various argument-based frameworks have been developed in defeasible reasoning [52, 56] for generating and evaluating arguments. Classically, argumentation has been mainly concerned with theoretical reasoning: reasoning about information attitudes such as knowledge and belief. Recently, a number of attempts have been made to use argumentation to capture practical reasoning: reasoning about what to do. This requires capturing arguments about pro-active attitudes, such as desires and intentions. In the following Subsection we discuss some of these relevant approaches.

2.5.1 Argumentation-based approaches to BDI agents

Some argument-based frameworks for practical reasoning are instantiations of Dung's abstract framework [52] (e.g. [3, 4]). Others are operational and grounded in logic programming (e.g. [134, 131, 132]).

Notably, a complete framework for practical reasoning is presented by Rahwan and Amgoud in [122]. This work is built on previous argumentation framework proposals for generating desires and plans [3, 4, 82, 91]. They provide a rich argumentation-based framework for (i) generating consistent desires, and (ii) generating consistent plans for achieving these desires. This is done through three distinct argumentation frameworks: one for arguing about beliefs, one for arguing about what desires the agent should adopt, and one for arguing about what plans to intend in order to achieve the agent's desires. More specifically, they refine and extend existing approaches by providing means for comparing arguments based on decision-theoretic notions (i.e. utility). Thus, the worth of desires and the cost of resources are integrated into the argumentation frameworks and taken into account when comparing arguments.

Recently, Amgoud and Prade in [6] has proposed an argumentation-based approach to formalize practical reasoning under uncertainty as a three steps process: 1) Deliberation which amounts to generate desires to be achieved, 2) Means-end reasoning which consists of generating compatible plans for achieving those desires and 3) Selecting the intentions to be pursued by the agent.

Another interesting argumentation approach to BDI agents is presented in [132]. In this work, they introduce a framework where defeasible argumentation is used for warranting agent's beliefs, filtering desires, and selecting proper intentions according to a given policy. In this framework, different types of agents can be defined and this decision will affect the way in which desires are filtered. The contribution of their approach is to introduce a BDI architecture that uses a concrete framework based on a working defeasible argumentation system: Defeasible Logic Programming (DeLP)

[60]. In DeLP, knowledge is represented using facts, strict rules, and defeasible rules. A Defeasible Logic Program (de.l.p.) is a set of facts, strict rules and defeasible rules. When required, $\mathcal{P}$ is denoted (Π, Δ) distinguishing the subset Π of facts and strict rules, and the subset Δ of defeasible rules.

In this framework [132] the main input is the *perception* from the environment, which is part of the set of the agent beliefs. Following [131], agent beliefs correspond to the semantics of a defeasible logic program $P_B = (\Pi_B, \Delta_B)$. In Π_B two disjoint subsets will be distinguished: Φ of perceived beliefs that will be updated dynamically, and Σ of strict rules and facts that will represent static knowledge, $\Pi_B = \Phi \cup \Sigma$. Then, besides the perceived beliefs, the agent may use strict and defeasible rules from P_B to obtain through an argumentation process the set B of warranted beliefs.

For selecting from the set D of possible desires, those that are suitable to be brought about, the agent uses its beliefs (representing the current situation) and a defeasible logic program (Π_F, Δ_F) composed of filtering rules. The filtering rules represent reasons for and against adopting desires. In other words, filtering rules eliminate those desires that cannot be effected in the situation at hand. Different agent types can be obtained depending on the chosen filtering criteria. Once the set of achievable desires is obtained (D^c), the agent can adopt one of them as an intention.

The final stage of this agent behaviour loop involves the usage of a set of *intention rules* to select the final set of intentions. The set of all applicable intention rules contains rules whose heads represent applicable intentions achievable in the current situation. Depending on the actual domain, there are many possible policies to be used. Then, using a suitable *intention policy* the agent will determine the preferred rule. The current desire in the head of this rule will be the selected intention. The existence of plans in order to satisfy a desire are related to intention rules and also, to the notion of achievable desire, but are not explicitly formalized.

This is a valuable argumentation-based proposal to BDI agents, with different argumentation frameworks for beliefs, desires and intentions. It has as limitation that it has been developed using bi-valued propositional formulae for representing belief, desires and intentions. A kind of uncertainty is represented by defeasible rules. As the Defeasible Logic Programming ($DeLP$) has recently been extended to an argumentation framework that includes the treatment of possibilistic uncertainty, named Possibilistic Defeasible Logic Programming ($PDeLP$) [2], we think that by using $PDeLP$ this approach can be extended to a graded argumentation-based approach to BDI agents.

Finally, in the following Subsection we describe in some detail the most relevant characteristics of the proposal due to Rahwan and Amgoud [122] because we found that this is the argumentation-based proposal that is closest to our work.

Rahwan and Amgoud's approach

We start by presenting the logical language used in this work, as well as the different data-bases representing the agent mental states and some special rules used in them.

From a propositional language $\mathcal{L}$ the agent can distinguish the three following sets of formulae: the set $\mathcal{K}$ which represents the agent knowledge, the set $\mathcal{D}$ which gathers all possible agent desires and the set RES which contains all the available resources in a system. From these sets, two kinds of rules can be defined: *desire-generation rules* and *planning rules*.

- *Desire-Generation Rule* or a desire rule, is an expression of the form $\varphi_1 \wedge ... \wedge \varphi_n \wedge \psi_1 \wedge ...\psi_m \Rightarrow \psi$ where $\varphi_i \in \mathcal{K}$ and $\psi_j, \psi \in \mathcal{D}$

The meaning of the rule is "if the agent believes $\varphi_1, ..., \varphi_n$ and desires $\psi_1, ..., \psi_m$, then the agent will desire ψ as well" and let

$head(\varphi_1 \wedge ... \wedge \varphi_n \wedge \psi_1 \wedge ...\psi_m \Rightarrow \psi) = \psi$

- *Planning Rule* is the basic building block for specifying plans and is an expression of the form

$\varphi_1 \wedge ... \wedge \varphi_n \wedge r_1 \wedge ...r_m \mapsto \varphi$ where $\varphi_i, \varphi \in \mathcal{D}$ and $\forall r_j \in RES$

A planning rule expresses that "if $\varphi_1 \wedge ... \wedge \varphi_n$ are achieved and the resources $r_1 \wedge ...r_m$ are used, then φ is achieved."

Let DGR and PR be the set of all possible desire generation rules and planning rules, respectively.

For the different argumentation processes the agent is equipped with four data-bases: $\langle B_b, B_d, B_p, R\rangle$,[1] where B_b base contains the agent beliefs, B_d its desire-generation rules, B_p its planning rules and R gathers all the agent resources.

- $B_b = \{(\beta_i, b_i),\ i = 1, ..., n\}$ where $\beta_i \in \mathcal{K}$ and $b_i \in [0, 1]$. A pair (β_i, b_i) means that the belief on β_i is at least b_i.
- $B_d = \{(dgr_i, w_i),\ dgr_i \in DGR\ and\ w_i \in \mathbb{R},\ i = 1, ..., m\}$ where symbol w_i denotes the worth of the desire $head(dgr)$.
- $B_p = \{pr_i,\ pr_i \in PR, i = 1, ..., l\}$
- $R = \{(r_i, c_i),\ i = 1, ..., n\}$ where $r_i \in RES$ and $c_i \in \mathbb{R}$ is the cost of consuming r_i.

Arguing over beliefs is different from arguing over desires or intentions. A proposition is believed because it is true and relevant. Desires, on the other hand, are adopted because they are justified and achievable. A desire is justified because the world is in a particular state that warrants its adoption. On the other hand, a desire

[1] B_i^* and R^* will denote the corresponding data-bases without degrees.

is achievable if the agent has a plan that achieves that desire. As a consequence of the different nature of beliefs and desires, they are supported by two different types of arguments. For example, a belief argument can be attacked by arguing that it is not consistent with observation, or because there is a reason to believe the contrary. Arguments for desires, on the other hand, could be attacked by demonstrating that the justification of that desire does not hold, or that the plan intended for achieving it is itself not achievable.

To deal with the different nature of the arguments involved, the authors present three distinct argumentation frameworks: one for reasoning about beliefs, another for arguing about what desires are justified and should be pursued, and a third for arguing about the best plan to intend in order to achieve these desires. The first framework for arguing about beliefs is based on existing works on belief argumentation [52] and can be seen in [122], next we describe the other argumentation schemas.

Arguing over desires

This proposal extends and refine the work by Amgoud and Kaci [4] considering more general desire-generation rules in the sense that a desire may not only be generated from beliefs, but it can also be generated from other desires.

In what follows, the functions BELIEFS(A), DESIRES(A) and CONC(A) return respectively, for a given argument A, the beliefs used in A, the desires supported by A and the conclusion of the argument A.

Let B_b and B_d the agent belief and desire bases, an *Explanatory Argument* is defined as follows:

- If $\exists(\Rightarrow \psi) \in B_d^*$ then $(\Rightarrow \psi)$ is an explanatory argument (A) with:
  BELIEFS(A)= $\emptyset$
  DESIRES(A)= $\{\psi\}$
  CONC(A) = ψ

- If $B_1, ..., B_n$ are belief arguments, and $E_1, ..., E_m$ are explanatory arguments, and exists
  $CONC(B_1) \wedge ... \wedge CONC(B_n) \wedge CONC(E_1) \wedge ... \wedge CONC(E_m) \Rightarrow \psi \in B_d^*$
  then $B_1, ..., B_n, E_1, ..., E_m \Rightarrow \psi$ is an explanatory argument (A) with:
  BELIEFS(A)= $SUPP(B_1) \cup ... \cup SUPP(B_n) \cup BELIEFS(E_1) \cup ... \cup BELIEFS(E_m)$
  DESIRES(A)= $DESIRES(E_1) \cup ... \cup DESIRES(E_m) \cup \{\psi\}$
  CONC(A) = ψ

Where the support of a belief argument $A = \langle H, h \rangle$ denoted by $SUPP(A)$, is a minimal consistent set of belief formulae H that infers h.

$TOP(A) = B_1, ..., B_n, E_1, ..., E_m \Rightarrow \psi$ is the top rule of the argument.

Let A_d denote the set of all explanatory arguments that can be generated from $\langle B_b, B_d \rangle$ and $A = A_d \cup A_b$. As with belief arguments, explanatory arguments may have different forces. However, since explanatory arguments involve two kinds of information: beliefs and desires, their strengths depend on both the quality of beliefs (using the notion of certainty level, i.e. the minimum certainty level of the support formulae) and the importance (weight) of the supported desire. Then, the pair (Level(A),Weight(A)) is used to define an order relation over explanatory arguments. Since beliefs verify the validity and the feasibility of desires, it is important that beliefs take precedence over desires. This is translated by the fact that the certainty level of the argument is more important than the weight of the desire.

The notion of attack is then defined, an explanatory argument for some desire can be defeated either by a belief argument (which undermines the truth of the underlying belief justification), or by another explanatory argument (which undermines one of the existing desires the new desire is based on).

The definition of acceptable explanatory arguments $Acc(A_d)$ is based on the notion of defense. An explanatory argument can be defended by either a belief argument or an explanatory argument. Desires supported by acceptable explanatory arguments are justified and hence the agent will pursue them (if they are achievable).

Arguing over plans

We have presented a framework for arguing about desires and producing a set of justified desires. Among this set, the agent must decide which ones will be pursued and with which plan. The basic building block of a plan is the notion of partial plan, which corresponds to a planning rule.

A complete plan (instrumental argument) is $\langle G, d \rangle$ such that $d \in D$ and G is a finite tree such that: the root of the tree is a partial plan for d; and a node has as many children as the premise of the partial plan has. For each desire φ_i there is a child node representing a partial plan toward this desire, and for each resource r_j there is an empty partial plan. The leaves of the tree are elementary partial plans (with null preconditions).

An instrumental argument may achieve one or several desires of different worth with a certain cost. So the strength of that argument is "the benefit" or "utility" defined in this approach as the difference between the worth of the desires and the cost of the plan. In a previous work [4] the authors defined the strength of an instrumental argument only on the basis of the weight of the corresponding desire and they did not account for the cost of executing the plan.

Let $A = \langle G, g \rangle$ be an instrumental argument, the utility of A is defined as:

$$Utility(A) = \sum_{d_i \in Des(G)} Worth(d_i) - \sum_{r_i \in Res(G)} Cost(r_i)$$

The notion of *conflict-free sets* of instrumental arguments is defined requiring consistency between each consistent set of agent beliefs and all the nodes in the

instrumental argument tree. Then, *acceptable sets of instrumental arguments* are defined as maximal conflict-free sets. Finally, a desire g is *achievable* iff exists an acceptable set of instrumental arguments S' such that $\langle G, g\rangle \in S'$

For an acceptable set of instrumental arguments S they define a global utility extending the difference to the sum of worth of the desires and the sum of costs to all the plans in S. This utility is used to construct a complete pre-ordering on the set of acceptable sets of instrumental arguments. The basic idea is to prefer the set of consistent plans with a maximal total utility.

This is more flexible than the frameworks presented in [3], where sets with maximal number of desires are privileged, with no regard to their priority or the cost of different plans. In order to be pursued, a desire should be both justified (i.e supported by an acceptable explanatory argument) and also achievable. Such desires will form the intentions of the agent.

Let $I \subseteq D$, I is an *intention set* iff:

1. $\forall d_i \in I,\ d_i$ is justified.

2. $\exists S_k$ acceptable set of instrumental arguments such that $\forall d_i \in I$, exists $\langle G_i, d_i\rangle \in S_k$.

3. $\forall S_j \neq S_k$ satisfying condition 2, then S_k is preferred to S_j.

4. I is maximal for set inclusion among the subsets of D satisfying the above conditions.

The second condition ensures that the desires are achievable together. If there is more than one intention set, a single one must be selected (e.g. at random) to become the agent's intention. This is a very complete argumentation-based approach to practical reasoning and we will show in Chapter 14 that it is complementary to our work in different aspects.

2.6 Conclusions

We conclude with some observations that will help us to outline our Thesis work. Without any doubt, the importance given to the use of multiagent technologies has increased in the design and implementation of complex real systems. In order to achieve the full potential of multiagent approaches there are some important challenges pointed in [102] for the next future. Some of them are (1) working on many different theories, architectures, technologies and infraestructures to specify, design, implement and manage agent based systems; (2) creating tools, techniques and methodologies to support agent systems developers and (3) establishing appropriate linkage with other branches of computes science and with other disciplines, like the uncertainty community in AI. In these general directions we place our research.

After a bibliographic review, we have noticed that there are interesting works dealing with partial aspects of graded attitudes in intentional systems (e.g., uncertainty in beliefs, graded or ordered desires, intention reconsideration in uncertain domains, etc), but we did not find many works proposing a general model. This has encouraged us to work on the extension of an agent intentional architecture to include graded attitudes.

Recently, Rahwan and Amgoud in [122] has presented an argumented-based approach to practical reasoning including three argumentation frameworks for beliefs, desires and plans. This is a valuable approach allowing to represent uncertain belief and worth related to desires. In this work, however, the authors do not present strictly speaking a formal system (in the sense of a logical system which is sound and complete with respect to an intended semantics) to represent and reason with these graded attitudes according to a suitable uncertainty model. We propose a complementary approach to practical reasoning, based on a multi-context framework that includes different logics to deal with these formal aspects and to include others, like some estimation on plan failure and more flexible rules to derive intentions.

Because its aforementioned relevance, we have chosen to deal with the BDI architecture. In particular, we have opted for a multi-context specification of the BDI model because this approach shortens the gap between specification and implementation, among other advantages. Besides, to represent and reason about the agent positive and negative desires in the g-BDI model we inspired our work in the bipolar model of preferences proposed by Benferhat et al. in [12].

Logic is the anatomy of thought.

J. Locke

Chapter 3

Logical Background

In this Chapter, we include the review of some logic backgrounds necessary for our agent model. First we revised the propositional Dynamic logic that allow us to represent and reason about the agent actions. Then, some many-valued logic as the Gödel logic expanded with a finite set of truth constants ($G_{\Delta}(C)$) and the Rational Pavelka logic (RPL) are presented. These logics will be used as the many-valued logic to reason about graded degrees in the different attitudes. Finally the Rational Lukasiewicz logic (RLL) is revised, this logic will be used in the agent intention formalization.

3.1 Propositional Dynamic logic

To define the PDL language, $\mathcal{L}_{DL}$, the propositional language $\mathcal{L}$ is thus extended by adding to it action modalities of the form $[\alpha]$ where α is an action. More concretely, given a set Π^0 of symbols representing elementary actions and the set $\mathcal{L}$ of basic formulae the set $\Pi(\mathcal{L}, \Pi^0)$ of plans (composite actions) is the following:

- $\Pi^0 \subset \Pi$ (elementary actions are plans)
- if $\alpha, \beta \in \Pi$ then $\alpha; \beta \in \Pi$, (the concatenation of actions is also a plan)
- if $\alpha, \beta \in \Pi$ then $\alpha \cup \beta \in \Pi$ (non-deterministic disjunction)
- if $\alpha \in \Pi$ then $\alpha^* \in \Pi$ (iteration)
- If $A \in \mathcal{L}$, then $A? \in \Pi$ (test)

then formulae $\mathcal{L}_{DL}$ is defined as follows:

- if $p \in Var$, then $p \in \mathcal{L}_{DL}$
- if $\varphi \in \mathcal{L}_{DL}$ then $\neg\varphi \in \mathcal{L}_{DL}$

- if $\varphi, \psi \in \mathcal{L}_{DL}$ then $\varphi \rightarrow \psi \in \mathcal{L}_{DL}$
- if $\alpha \in \Pi$ and $\varphi \in \mathcal{L}_{DL}$ then $[\alpha]\varphi \in \mathcal{L}_{DL}$.

The interpretation of $[\alpha] A$ is *"after the execution of α, A is true"*. We denote by $\vdash_{DL}$ the notion in proof in DL.

The semantics for the language $\mathcal{L}_{DL}$ is defined, as usual in modal logics, using a Kripke structure.

A *standard Kripke model* is a structure $\langle W, \{R_\alpha : \alpha \in \Pi\}, e\rangle$ where

- W is a non-empty set of possible worlds.
- $R_\alpha \subseteq W \times W$, for each $\alpha \in \Pi$. This relation represents transition over worlds by the execution of action α.
- $e : V \times W \rightarrow \{0, 1\}$ provides for each world a Boolean (two-valued) evaluation of the propositional variables, that is, $e(p, w) \in \{0, 1\}$ for each propositional variable $p \in V$ and each world $w \in W$ and the evaluation is extended to arbitrary formulae in $\mathcal{L}_{DL}$ using classical connectives and to formulae with action modalities —as $[\alpha] A$, by defining:

  $e([\alpha] A, w) = min \{e(A, w') \mid (w, w') \in R_\alpha\}$

  Notice that $e([\alpha] A, w) = 1$ iff the evaluation of A is 1 in all the worlds w' that may be reached through the action α from w.

Then, a *regular Kripke model* is a standard Kripke model where the R relation also verify:

- $R_{\alpha;\beta} = R_\alpha \circ R_\beta$,
- $R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$,
- $R_{\alpha^*} = (R_\alpha)^*$ (ancestral relation) and
- $R_{\varphi?} = \{(w, w) \mid e(\varphi, w) = 1\}$, $\varphi \in \mathcal{L}$.

The simple propositional dynamic logic PDL introduced above can be finitely axiomatized by the following system [104]:

- any axiomatization of propositional logic.
- $[\alpha] (\phi \rightarrow \varphi) \rightarrow ([\alpha] \phi \rightarrow [\alpha] \varphi)$
- $[\varphi?] \phi \leftrightarrow \varphi \rightarrow \phi$

- $[\alpha;\beta]\,\phi \leftrightarrow [\alpha]\,[\beta]\,\phi$
- $[\alpha \cup \beta]\,\phi \leftrightarrow [\alpha]\,\phi \wedge [\beta]\,\phi$
- $[\alpha^*]\,\phi \rightarrow \phi$
- $[\alpha^*]\,\phi \rightarrow [\alpha]\,[\alpha^*]\,\phi$
- $[\alpha^*]\,(\phi \rightarrow [\alpha]\,\phi) \rightarrow (\phi \rightarrow [\alpha^*]\,\phi)$
- $[\alpha]\,\phi \leftrightarrow \neg\,[\alpha^*]\,\neg\phi$

Rules: modus ponens (MP) and the Necessitation rule: from φ derive $[\alpha]\,\varphi$.

3.2 $G_\Delta(C)$ and RPL Fuzzy Logics

Most of the many-valued systems related to fuzzy logic are those corresponding to *t-norm based fuzzy logics*. They use the real interval [0, 1] as set of truth-values and their calculi is defined by a conjunction & and an implication $\Rightarrow$ interpreted respectively by a t-norm $*$ and its residuum $\Rightarrow$ [1], and where negation is defined as $\neg\varphi = \varphi \rightarrow \bar{0}$, with $\bar{0}$ being the truth-constant for falsity. In the framework of these logics, called t-norm based fuzzy logics, each (left continuous) t-norm $*$ uniquely determines a semantical (propositional) calculus PC(*) over formulae defined in the usual way from a countable set of propositional variables, connectives $\wedge$, & and $\rightarrow$ and truth-constant $\bar{0}$. Evaluations of propositional variables are mappings e assigning each propositional variable p a truth-value $e(p) \in [0,1]$, which extend univocally to compound formulae as follows:

$$\begin{aligned} e(\bar{0}) &= 0 \\ e(\varphi \wedge \psi) &= min(e(\varphi), e(\psi)) \\ e(\varphi\&\psi) &= e(\varphi) * e(\psi) \end{aligned}$$

Note that, by definition of residuum, $e(\varphi \rightarrow \psi) = 1$ iff $e(\varphi) \leq e(\psi)$, in other words, the implication $\rightarrow$ captures the ordering. Further connectives are defined as follows:

$$\begin{aligned} \neg\varphi &= \varphi \rightarrow \bar{0} \\ \varphi \vee \psi &= ((\varphi \rightarrow \psi) \rightarrow \psi) \wedge ((\psi \rightarrow \varphi) \rightarrow \varphi) \\ \varphi \equiv \psi &= (\varphi \rightarrow \psi)\&(\psi \rightarrow \varphi) \end{aligned}$$

From the above definitions: $e(\varphi \vee \psi) = max(e(\varphi), e(\psi))$, $\neg\varphi = e(\varphi) \Rightarrow 0$ and $e(\varphi \equiv \psi) = e(\varphi \vee \psi) * e(\psi \rightarrow \varphi)$.

[1] Defined as $x \Rightarrow y = max\,\{z \in [0,1] \mid x * z \leq y\}$

A formula φ is a said to be a 1-tautology of PC(*) if $e(\varphi) = 1$ for each evaluation e, and will be denoted as $\models_* \varphi$. The associated consequence relation is defined as usual: if T is a theory (set of formulas), then $T \models_* \varphi$ whenever $e(\varphi) = 1$ for all evaluations e such that $e(\psi) = 1$ for all $\psi \in T$. Two relevant examples of continuous T-norm fuzzy logic are Gödel and Lukasiewicz logics defined by the following operations:

Gödel logic calculus:

$$\begin{aligned} x *_G y &= min(x, y) \\ x \Rightarrow_G y &= \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{otherwise} \end{cases} \end{aligned}$$

Lukasiewicz logic calculus:

$$\begin{aligned} x *_L y &= max(0, x + y - 1) \\ x \Rightarrow_L y &= \begin{cases} 1, & \text{if } x \leq y \\ 1 - x + y, & \text{otherwise} \end{cases} \end{aligned}$$

Notice that in these two calculi the min operation and hence the connective $\wedge$, is also definable from $*$ and $\rightarrow$ as:

$$min(x, y) = x * (x \Rightarrow y)$$

If we denote by $\vdash_L$ and $\vdash_G$ the provability relations in Lukasiewicz and Gödel logic respectively, the standard completeness hold, namely:

$$\begin{aligned} \vdash_L \varphi \text{ iff } \models_L \varphi \\ \vdash_G \varphi \text{ iff } \models_G \varphi \end{aligned}$$

Both logics have been shown to be axiomatic extensions of Hájek's Basic Fuzzy logic BL [78]. The following formulae are axioms of BL:

(A1) $(\varphi \rightarrow \psi) \rightarrow ((\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi))$

(A2) $(\varphi \& \psi) \rightarrow \varphi$

(A3) $(\varphi \& \psi) \rightarrow (\psi \& \varphi)$

(A4) $(\varphi \& (\varphi \rightarrow \psi) \rightarrow (\psi \& (\psi \rightarrow \varphi))$

(A5a) $(\varphi \rightarrow (\psi \rightarrow \chi) \rightarrow (\varphi \& \psi) \rightarrow \chi)$

(A5b) $(\varphi \& \psi) \rightarrow \chi) \rightarrow (\varphi \rightarrow (\psi \rightarrow \chi))$

(A6) $(\varphi \rightarrow \psi) \rightarrow \chi) \rightarrow (((\psi \rightarrow \varphi) \rightarrow \chi) \rightarrow \chi)$

(A7) $(\bar{0} \rightarrow \varphi)$

The deduction rule of BL is modus ponens.

Particularly, Lukasiewicz logic is an extension of BL by the axiom:

(L) $\neg\neg\varphi \to \varphi$ (forcing the negation to be involutive)

and Gödel logic adds to BL axiomatic the following axiom:

(G) $\varphi \to \varphi\&\varphi$ (forcing the conjunction to be idempotent)

The above mentioned completeness for theorems extend to deductions from arbitrary theories in case of Gödel logic and only to deductions from finite theories in case of Lukasiewicz logic:

$$T \vdash_L \varphi \text{ iff } T \models_L \varphi, \text{ if T is finite}$$
$$T \vdash_G \varphi \text{ iff } T \models_G \varphi$$

In some situations one might be also interested to explicitly represent and reason with partial degrees of truth. One convenient way to allow for an explicit treatment of degrees of truth is by introducing truth-constants into the language i.e. new constant symbols $\bar{c}$ for suitable values $c \in [0,1]$ and stipulates that $e(\bar{c}) = c$ for all truth-evaluations e, then a formula of the kind $\bar{c} \to \varphi$ becomes 1-true under any evaluation e whenever $c \leq e(\varphi)$.

Rational Pavelka Logic (RPL) is the expansion of Lukasiewicz Logic by adding into the language a truth-constant $\bar{r}$ for each rational $r \in [0,1]$, together with a number of additional axioms. The semantics is the same as Lukasiewicz logic, just expanding the evaluations e of propositional variables in $[0,1]$ to truth-constants by requiring $e(\bar{r}) = r$. Pavelka proved that his logic is complete for arbitrary theories in a non-standard sense known as Pavelka-style completeness. Namely, he defined the truth degree of a formula φ in a theory T as follows:

$$\|\varphi\|_T = \inf\{e(\varphi) \mid e \text{ is a RPL model of } T\}$$

and defined a provability degree of φ over T as:

$$|\varphi|_T = \sup\{r \mid T \vdash_{RPL} \bar{r} \to_L \varphi\}$$

and proved that $\|\varphi\|_T = |\varphi|_T$

Hájek showed [78] that in order to set an axiomatic for this logic only was necessary to add Lukasiewicz logic axioms these *book-keeping axiom* schemas:

$$\bar{r}\&\bar{s} \leftrightarrow \overline{r *_L s}$$
$$\bar{r} \to \bar{s} \leftrightarrow \overline{r \Rightarrow_L s}$$

On the other hand, Hájek also shows that Gödel logic can be expanded with a finite set of truth constants together with a new unary connective Δ (denoted $G_\Delta(C)$ logic) while preserving the strong standard completeness. Namely, let $C \subseteq [0,1]$ a finite set containing 1 and 0, and introduce into the language a truth-constant r for each $r \in C$, together with the so-called Baaz's projection connective Δ. Truth-evaluations of Gödel logic are extended in an analogous way to RPL as it regards to truth constants and adding the clause

$$e(\Delta(\varphi)) = \begin{cases} 1, & \text{if } e(\varphi) = 1 \\ 0, & \text{otherwise} \end{cases}$$

The axioms and rules for the $G_\Delta(C)$ logic are those of Gödel logic plus the above book-keeping axioms for constants in C and the following axioms and rules for Δ:

(Δ1) $\Delta\varphi \vee \neg\Delta\varphi$

(Δ2) $\Delta(\varphi \vee \psi) \to (\Delta\varphi \vee \Delta\psi)$

(Δ3) $\Delta\varphi \to \varphi$

(Δ4) $\Delta\varphi \to \Delta\Delta\varphi$

(Δ5) $\Delta(\varphi \to \psi) \to (\Delta\varphi \to \Delta\psi)$

and the Necessitation rule for Δ: from φ derive $\Delta\varphi$. Then, the following *strong completeness* result holds: for any theory T and formula φ,

$$T \vdash_{G_\Delta(C)} \varphi \text{ iff } T \models_{G_\Delta(C)} \varphi$$

3.2.1 Rational Lukasiewicz Logic

Rational Łukasiewicz logic, RLL for short, introduced by Gerla in [67] to represent and axiomatize the semantics of degrees of the intentions. This logic is an expansion of Łukasiewicz logic with a countable set of unary connectives $\{\delta_n\}_{n\in\mathbb{N}}$, whose intended semantics is that the truth-value of $\delta_n\varphi$ is just the truth-value of φ divided by n. So in RLL one can express divisions by natural numbers

The RLL language is an extension of Lukasiewicz Logic by adding into the language a truth-constant $\bar{r}$ for each rational $r \in [0,1]$) and the unary connectives δ_n for division, i.e. the language also includes the formulae $\delta_n\Phi$ if $\Phi \in \mathcal{L}_L$.

The following axioms defining the behavior of δ_n connectives, are added to the RPL axiomatics:

(δ_n1) $\delta_n\Phi \oplus \overset{n}{\ldots} \oplus \delta_n\Phi \equiv_L \Phi$

(δ_n2) $\delta_n\Phi \otimes (\delta_n\Phi \oplus \overset{n-1}{\ldots} \oplus \delta_n\Phi) \to_L \overline{0}$

As in Łukasiewicz logic the standard completeness holds and also Pavelka-style completeness holds for arbitrary theories [67].

Part II

The Graded BDI Agent Model

Chapter 4

The General Framework

4.1 Introduction

Several previous works have proposed theories and architectures to provide multiagent systems with a formal support. Among them, one of the most widely used is the BDI agent architecture presented by Rao and Georgeff. We consider that an extension of this architecture in order to incorporate degrees in the different attitudes, will not only make the model semantics richer, but it also will help the agent to take better decisions. With that aim we looked first at the "individual" aspect of agency, and decided to extend the BDI agent architecture to represent and reason under uncertain beliefs and graded motivations. In this Chapter we introduce a general model for graded BDI agents (g-BDI). This model is based on a multi-context specification of agents and is able to represent graded mental attitudes. In this sense, belief degrees will represent to what extent the agent believes a formula is true. Degrees of positive or negative desire shall allow the agent to set different levels of preference or rejection respectively. Intention degrees shall give also a preference measure but, in this case, modeling the cost/benefit trade off of reaching an agent's goal. Then, Agents having different kinds of behavior shall be modelled on the basis of the representation and interaction of these basic three attitudes.

In order to represent and reason about graded notions of beliefs, desires and intentions in our graded BDI agent model, we decide to use the many-valued modal approach proposed by Hájek et al. in [78]. Following this approach to reason about uncertainty in the different mental contexts, respecting a particular uncertainty model (e.g. probabilities, necessities), can be done in a very elegant way within a uniform and flexible logical framework.

In this Chapter we present the general aspects of this agent model as its multi-context specification and its logical framework. Then, to complete the definition of this agent model the specification of the different components in the architecture (i.e., the different contexts and bridge rules) are needed and will be presented in next Chapters 5, 6 and 7. The architecture we present will serve as a blueprint to design different kinds of particular agents.

4.2 Multi-context specification

The architecture presented in this Chapter is an extension of the work of Parsons et.al. [115] about multi-context BDI agents. Multi-context systems were introduced by Giunchiglia et.al. [71] to allow different formal (logic) components to be defined and interrelated. These systems and their applications to multiagent formalization were presented in Chapter 2.3.

The MCS specification of an agent contains three basic components: units or contexts, logics, and bridge rules, which channel the propagation of consequences among theories. Thus, an agent is defined as a group of interconnected units: $\left\langle \{C_i\}_{i\in I}, \Delta_{br} \right\rangle$, where each context $C_i \in \{C_i\}_{i\in I}$ is the tuple $C_i = \langle L_i, A_i, \Delta_i \rangle$ where L_i, A_i and Δ_i are the language, axioms, and inference rules respectively. They define the logic for the context and its basic behaviour as constrained by the axioms. When a theory $T_i \in L_i$ is associated with each unit, the implementation of a particular agent is complete. Δ_{br} can be understood as rules of inference with premises and conclusions in different contexts, for instance:

$$\frac{C_1 : \psi, C_2 : \varphi}{C_3 : \theta}$$

means that if formula ψ is deduced in context C_1 and formula φ is deduced in context C_2 then formula θ is added to context C_3.

The deduction mechanism of these systems is based on two kinds of inference rules, internal rules Δ_i inside each unit, and bridge rules Δ_{br} outside. Internal rules allow to draw consequences within a theory, while bridge rules allow to embed results from a theory into another [68]. A multi-context systems needs some kind of control strategy to synchronize and coordinate both kinds of inferences (i..e. by internal rules and by bridge rules) also needs to prevent that a bridge rule execute more than once under the same premise conditions.

We have *mental* contexts to represent beliefs (BC), desires (DC) and intentions (IC). We also consider two *functional* contexts: for Planning (PC) and Communication (CC). The Planner is in charge of finding plans to change the current world into another world, where some desire is satisfied, and of computing the cost associated to the plans. The Communication context is the agent door to the external world, receiving and sending messages. In summary, the BDI agent model is defined as:

$$A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br})$$

Each context has an associated logic, that is, a logical language with its own semantics and deductive system. The different context will be described in some detail in the following chapters.

In Figure 11.6 it is shown an schema of the graded BDI agent proposed make up of a set of mental contexts (BC, DC and DC) and functional ones (PC and CC)

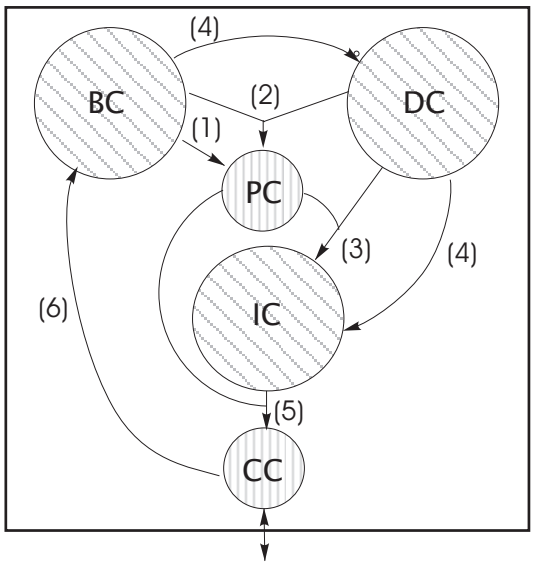

Figure 4.1: Multi-context model of a graded BDI agent

and some of the bridge rules ((1) to (6)) relating them.

4.3 Logical Framework: Many-valued modal approach

In the last two decades, the Artificial Intelligence community has undertaken the problem of knowledge representation and reasoning under uncertain, incomplete and vague knowledge. This was an important and necessary issue, in order to develop systems able to deal with these kinds of information in real-domains. There are different approaches to model and manage approximate reasoning. Among the most relevant ones, are the works based on probabilistic models, Dempster-Shafer theory of evidence and possibility theory [100]. Considering uncertainty reasoning an important state of the art can be seen in [79].

Recently, Hájek et al. in e.g. [78] and Gottwald in [75] have developed an approach where uncertainty reasoning is dealt by defining suitable modal theories over suitable many-valued logics. This proposal allows to use well-founded logical frameworks (as diverse many-valued logics) to represent different uncertainty models by adding the adequate axiomatics for each case.

Fuzzy logics and uncertainty theories play different roles that must be clarified. Fuzzy logic is a logic of vague, imprecise notions and propositions, and is then, a logic of partial degrees of truth. On the contrary, an uncertainty measure deals with crisp notions and propositions (i.e. true or false), and is evaluated with the degree of belief on the truth of the proposition. Fuzzy logics behave as a many-valued logic,

whereas uncertainty or belief theories can be related to some kinds of (two-valued) modal logics.

Next, we present the basic ideas proposed by Hájek et al. in [78]. We notice that they are expressed in terms of beliefs but they may be directly applied to other graded mental attitudes (e.g. desires, intentions).

To consider the belief degree of a crisp proposition as the truth-degree of a fuzzy (modal) proposition.

For instance, let us consider the case where belief degrees are modelled as probabilities. Then, for each classical (two-valued) formula φ, we consider a modal formula $B\varphi$ which is interpreted as "φ is probable". This modal formula $B\varphi$ may be seen then as a *fuzzy* formula which may be more or less true, depending on the probability of φ. In particular, we can take as truth-value of $B\varphi$ precisely the probability of φ. Moreover, using a many-valued logic, we can express the governing axioms of probability theory as logical axioms involving modal formulae of the kind $B\varphi$. But notice that such an approach has to clearly distinguish between propositions like "(ψ is probable) and (φ is probable)" on the one hand and "$(\psi \wedge \varphi)$ is probable" on the other.

Other key concept pointed out in the works by Godo et. al [72, 73] related to reason about belief functions using fuzzy logic is:

To represent belief measures as models of fuzzy theories.

Once we have defined the language of belief fuzzy propositions $B\varphi$ (B being a modality representing probable, necessary, desirable, etc.) where φ are crisp propositions, then we can write theories about the $B\varphi$ formulae over a particular fuzzy logic.

Then, the many-valued logic machinery can be used to reason about the modal formulae $B\varphi$, which faithfully respect the uncertainty model chosen to represent the degrees of belief. Therefore, in this kind of logical frameworks we shall have, besides the axioms of the many-valued logic, a set of axioms corresponding to the basic postulates of a particular uncertainty theory.

The question derived from this approach is which kind of fuzzy logics can be used and which aspects of uncertainty theories can be formalized. As for example, Hájek et al. in [77] defined a propositional probability logic —Fuzzy Logic of Probability, as a theory over Rational Pavelka logic *RPL* (an extension of Łukasiewicz's infinitely-valued logic with rational truth constants, described in Section 3.2). The very reason of selecting this logic was in first place, the availability of well-founded results for this logic and secondly, the fact that the main connectives of this logic are based on the arithmetic addition in [0,1], which is basically what we need to deal with additive measures like probabilities. Besides, Łukasiewicz logic also allows to define the *min* and *max* connectives, so it was possible to define in [77] a logic to reason about

necessities and possibilities degrees. However, in order to get standard completeness for a necessity-based logic, instead of Pavelka-style completeness, it may be chosen the $G_\Delta(C)$ logic (see for details Section 3.2), because, the main connective is the *min* conjunction which is vital to represent necessity measures, since necessity measures are min-decomposable w.r.t. the conjunction $\wedge$, and second, because $G_\Delta(C)$ enjoys completeness for deductions from arbitrary theories.

On the other hand, considering the Łukasiewicz logic as the underlying fuzzy logic constraints the uncertainty theories that can be defined over it. In particular, the lacking of the product and division operations is an important obstacle to formalize many aspects of uncertainty theories. Then, the same authors in [72] take advantage of a logic combining Łukasiewicz and Product logic connectives —ŁΠ Logic— to define a richer belief theory on top of it, particularly they formalized the logic of conditional probability.

To give an insight of how these logical frameworks are built, we consider some features of the Fuzzy Logic of Probability (FP) presented in [77]. They associate with each crisp formula ψ a new *fuzzy propositional variable* $P(\psi)$ meaning "ψ is probable" and which is evaluated using its probability: $e(P\psi) = P(\psi)$.

The syntax of FP-formulae are just *RPL*-formulae built from *fuzzy propositional variables*. Then the FP-logic language includes formulae of two types, namely:

1. **non-modal:** they are crisp formulae of $\mathcal{L}$ i.e., those built from propositional variables $p_1, ..., p_n$ using the classical binary connectives ($\neg$, $\vee$).

2. **modal:** they are built from elementary modal formulae $P\psi$ (also may be noted f_ψ), were ψ are non-modal formulae, and the truth constants $\bar{r}$ for each rational $r \in [0, 1]$ using the connectives of RPL ($\&, \rightarrow_L, \leftrightarrow_L, \wedge, \vee$)

The axiomatic schema presented for the FP-logic is the following:

- (RPL) Axioms of *RPL. (Axioms of many-valued logic)*
  (can be seen in Section 3.2)

- (FP1) $(P\phi, 1)$ for ϕ being an axiom of classical propositional logic.
  Necessitation rule

- (FP2) $(P(\phi \rightarrow \psi) \rightarrow_L (P\phi \rightarrow_L P\psi), 1)$ for all ϕ, ψ
  (K axiom for the P modality)

- (FP3) $(P\neg\phi \rightarrow_L \neg_L(P\phi), 1)$ for ϕ for all ϕ
  (Axiom of probability)

- (FP4) $(P(\phi \vee \psi) \leftrightarrow_L ((P\phi \rightarrow_L P(\phi \wedge \psi)) \rightarrow_L P\psi), 1)$ for all ϕ, ψ
  (Axiom of probability)

Axioms (FP1) and (FP2) guarantee the preservation of classical equivalence and the monotonicity. (FP3) and (FP4) are direct translation of the well-known axiom of probability, the first represents the relationship between the probability of one proposition and its negation (i.e. $P\neg\phi = 1 - P\phi$), and the second represents the finitely additive property (i.e. $P(\phi \vee \psi) = P\phi + P\psi - P(\phi \wedge \psi)$).

In order to represent and reason about graded notions of beliefs, desires and intentions in our graded BDI agent model, we decide to use the many-valued modal approach previously presented. Following this approach to reason about uncertainty, respecting a particular model (e.g. probabilities, necessities), can be done in a very elegant way within a uniform and flexible logical framework.

The matter is to select the suitable many-valued logic that equip us with the necessary connectives as to represent the selected uncertainty measure. In this way, the most suitable many-valued logic with well-founded results, may be used as the logic basement for the uncertainty theory we want to model. Furthermore, the logic machinery of this many-logic can be used to reason about the uncertainty measure.

In our g-BDI model, for the probabilistic model of the belief context and for the desire and intention contexts, we choose as the many-valued logic the infinite-valued Łukasiewicz logic.[1] But another selection of many-valued logics may be done for each unit according to the uncertainty measure modelled in each case. For instance, we use Gödel logic for the necessity approach to the BC because the main connective is the *min* conjunction which is fundamental to represent necessity measures, and also because $G_\Delta(C)$ logic presents completeness for deductions from arbitrary theories.

Therefore, in this kind of logical frameworks we shall have, besides the axioms of many-valued logic (e.g Łukasiewicz or Gödel logics), a set of axioms corresponding to the basic postulates of a particular uncertainty theory. Hence, in this approach, reasoning about uncertainty (e.g. probabilities, necessities) can be done in a very elegant way within an uniform and flexible logical framework.

This many-valued logical framework is used as the general logical framework to represent and reason about the different mental graded attitudes in the g-BDI agent model, as will be seen in Chapters 5 and 6.

[1]The reason of using this many-valued logic is that its main connectives are based on the arithmetic addition in the unit interval $[0, 1]$, which is what is needed to deal with additive measures like probabilities. Besides, Łukasiewicz logic has also the *min* conjunction and *max* disjunction as definable connectives, so it also allows to define a logic to reason about degrees of necessity and possibility.

So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality.

A. Einstein

Chapter 5

The Belief Context (BC)

5.1 Introduction

The purpose of this context is to model the agent's beliefs about the environment. These beliefs may be uncertain, imprecise or incomplete. In the last years, to overcome with this kind of information, different models of approximate reasoning have been proposed to represent and reason with these kinds of knowledge [79]. Among the main approaches to uncertainty we can mention the probabilistic, possibilistic and evidential models.

In the BC we represent the agent uncertain knowledge about the world where she lives. We need quantitative formulae to express, for instance, that the belief degree (e.g. probability, necessity) of a formula is greater or equal than a certain value. Also, qualitative formulae are needed to represent that the belief degree in some formula is greater than the belief in other. To represent these beliefs in the Belief context we use modal many-valued formulae, following the logical framework presented in Section 4.3.

Since the agent needs to reason about her possible actions and the environment transformations they cause, they must be part of any situated agent's beliefs set. To represent this knowledge related to action execution, we use the Dynamic Propositional logic (PDL) as the basic propositional logic (described in Section 3.1). PDL has been proposed to model agent's actions in [104] and [140].

We consider two particular approaches to uncertainty for the Belief Context: BC_{nec} corresponding to the necessity-valued logic and BC_{prob} using the probability-valued logic. For each logic, a modal language BC is defined over a propositional dynamic language to reason about the belief on dynamic propositions. Then, our logics BC_{nec} and BC_{prob} are built as suitable theories over suitable fuzzy modal logics with truth-constants. Particularly, the necessity logic for the BC (BC_{nec}) is defined as a theory over $G_{\Delta}(C)$ whereas the probabilistic logic for BC (BC_{prob}) is formalized as a theory in Rational Pavelka logic (RPL). The description of both fuzzy logics, $G_{\Delta}(C)$ and RPL, can be seen in Chapter 3.

Furthermore, a sound and complete axiomatics for BC_{nec} and BC_{prob} is presented.

Other belief models might be used as well selecting an appropriate many-valued logic and adding the corresponding axioms.

5.2 Belief context logics

To reason about the credibility of bi-valued (crisp) propositions, we define a language for belief representation, following Godo et al.'s proposal [72, 73] based on a many-valued logic (described in Section 4.3).

In order to define the basic crisp language, we start from a classical propositional language $\mathcal{L}$, defined upon a countable set of propositional variables Var and connectives $(\neg, \rightarrow)$, and extend it to represent actions. We take advantage of Dynamic logic which is suitable to model agent's actions (described in Section 3.1). These actions and the transformations caused by their execution must be part of any situated agent's beliefs set.

The propositional language $\mathcal{L}$ is thus extended to $\mathcal{L}_{DL}$, by adding to it action modalities of the form $[\alpha]$ where α is an action. The set of composite actions or plans Π are built from the language $\mathcal{L}$ and a given set Π^0 of symbols representing elementary actions. Then, $\mathcal{L}_{DL}$ includes formulae like $[\alpha]\,\varphi$, where $\alpha \in \Pi$ and $\varphi \in \mathcal{L}$. The definition of the plan set Π and the propositional dynamic language $\mathcal{L}_{DL}$ can be seen in Section 3.1.

This language $\mathcal{L}_{DL}$ is the basic bi-valued (crisp) language to define a fuzzy modal language capable to represent the uncertainty of the belief context. Once we build a belief fuzzy proposition $B\varphi$ (B being a modality standing for probable, necessary, believable, etc.) per each crisp proposition φ, then one can try to write theories about formulae of the form $B\varphi$ over a particular fuzzy logic, including, as axioms, formulae corresponding to the basic postulates of a particular uncertainty theory. In this way, models (in the sense of many-valued logic) of the theories about formulae of the form $B\varphi$ become uncertainty measures of a particular type over the crisp formulae φs.

Formulae of BC are of two types:

- *Crisp (non B-modal):* they are the (crisp) formulae of $\mathcal{L}_{DL}$, built in the usual way, thus, if $\varphi \in \mathcal{L}_{DL}$ then $\varphi \in BC$.
- *B-Modal:* they are built from elementary modal formulae $B\varphi$, where φ is crisp, and truth constants $\overline{r}$, for each rational $r \in [0, 1]$, using the connectives of the chosen many-valued logic.

Now, in order to represent and reason about the uncertainty of dynamic formulae we need to introduce appropriate measures for this purpose. Necessity and probability measures are two outstanding plausibility measures [79].

Given a Boolean algebra F of subsets of W, $\mu : F \to [0,1]$ is a plausibility measure if the following holds:

- $\mu(\emptyset) = 0$
- $\mu(W) = 1$
- If $X, Y \in F$ and $X \subseteq Y$, then $\mu(X) \leq \mu(Y)$

A plausibility measure μ is a

1. *necessity measure* if in addition satisfies:
   - $\mu(X \cap Y) = \min(\mu(X), \mu(Y))$
2. *(finitely additive) probability* if it satisfies:
   - $\mu(X \cup Y) = \mu(X) + \mu(Y)$, when $X \cap Y = \emptyset$, for all $X, Y \subseteq F$

We define next two approaches for the Belief Context: BC_{nec} corresponding to the necessity-valued logic and BC_{prob} using the probability-valued logic. For each logic, a modal language BC is defined over the language $\mathcal{L}_{DL}$ to reason about the belief on crisp dynamic propositions. To do so, we extend the crisp language $\mathcal{L}_{DL}$ with a fuzzy unary modal operator B. If φ is a proposition in $\mathcal{L}_{DL}$, the intended meaning of $B\varphi$ is "φ is certain" in one case and "φ is probable" in the other. Then, our logics BC_{nec} and BC_{prob} are built over the set $\mathcal{L}_{DL}$ of DL-formulae, as theories over suitable fuzzy logics with truth-constants.

Now, the question is which kind of fuzzy logics can we use and which (aspects of) uncertainty theories can we formalize. The very reason of using Rational Pavelka logic (RPL) for a probabilistic model as in [72] is, besides the availability of well-founded results for such a logic, the fact that the main connectives of Łukasiewicz's logic are based on the arithmetic addition in the unit interval [0, 1], which is obviously what we basically need to deal with additive measures like probabilities. Also we have the Pavelka style completeness for this logic, that does not need finite theories, which fits our needs when proving completeness for the BC_{prob} logic.

For a necessity model of uncertainty, the Łukasiewicz's logic also has the *min* conjunction and *max* disjunction as definable connectives, and it is not difficult to also define a logic to reason about necessity or possibility degrees. However, in order to get standard completeness for our necessity-based BC logic, instead of Pavelka-style completeness, we chose the $G_\Delta(C)$ logic, because, first the main connective is the *min* conjunction which is vital to represent necessity measures, since necessity measures are min-decomposable w.r.t. the conjunction $\wedge$, and second, because $G_\Delta(C)$ enjoys completeness for deductions from arbitrary theories that we can extend to our Belief context logic BC_{nec}.

Therefore, the necessity logic for the BC will be defined as a theory over $G_\Delta(C)$ and will be denoted BC_{nec}, whereas the probabilistic logic for BC will be defined as a theory in Rational Pavelka logic RPL and will be denoted BC_{prob}.

5.3 The BC_{nec} Logic

We define a fuzzy modal language over the DL-formulae to reason about the necessity degree of dynamic formulae. In this case, we chose as many-valued logic: $G_{\Delta}(C)$, i.e. Gödel logic expanded with the Δ operator and *finitely many* truth-constants from some finite $C \subset [0,1]$ [78].

The language of BC_{nec}, $\mathcal{L}_{BCn}$, is built over the language $\varphi \in \mathcal{L}_{DL}$. Then, formulae of $\mathcal{L}_{BCn}$ are of two types DL-formulae and B-modal formulae (B-formulae), namely:

- If $\varphi \in \mathcal{L}_D$ then $\varphi \in \mathcal{L}_{BCn}$.
- If $\varphi \in \mathcal{L}_D$ then $B\varphi \in \mathcal{L}_{BCn}$
- If $r \in C$ then $\overline{r} \in \mathcal{L}_{BCn}$
- If $\Phi \in \mathcal{L}_{BCn}$ then $\Delta(\Phi) \in \mathcal{L}_{BCn}$ (where Δ represents the unary Baaz projection connective).
- If $\Phi, \Psi \in \mathcal{L}_{BCn}$ then $\Phi \rightarrow_G \Psi \in \mathcal{L}_{BCn}$ and $\Phi \wedge_G \Psi \in \mathcal{L}_{BCn}$ (where $\wedge_G$ and $\rightarrow_L$ correspond respectively to the conjunction and implication of $G_{\Delta}(C)$ logic). Other $G_{\Delta}(C)$ logic connectives are definable from $\wedge_G$, $\rightarrow_G$, and $\overline{0}$: $\varphi \vee_G \psi$ is $((\varphi \rightarrow_G \psi) \rightarrow_G \psi) \wedge_G ((\psi \rightarrow_G \varphi) \rightarrow_G \varphi))$, $\varphi \equiv_G \psi$ is $(\varphi \rightarrow_G \psi) \wedge_G (\psi \rightarrow_G \varphi)$, and $\neg_G \varphi$ is $\varphi \rightarrow_G \overline{0}$.

Since in Gödel logic a formula $\Phi \rightarrow_G \Psi$ is 1-true iff the truth value of Ψ is greater or equal to that of Φ, modal formulae of the type $\overline{r} \rightarrow_G B\varphi$ (where $\varphi \in \mathcal{L}_{DL}$) express that the necessity of φ is at least r (quantitative formulae). By means of formulae of the form $B\psi \rightarrow_G B\varphi$ comparison of degrees may be represented (qualitative formulae).

In this context, the agent's beliefs will be expressed by a theory $\mathcal{T}$ (a set of B-formulae) containing quantitative and qualitative expressions.

BC_{nec} semantics

The semantics for this language is defined using a Necessity Kripke structure of the following form:

$$M_{BCn} = \langle W, \{R_\alpha : \alpha \in \Pi\}, e, \mu \rangle$$

where $\langle W, \{R_\alpha : \alpha \in \Pi\}, e \rangle$ is a regular Kripke model of PDL and $\mu : F \rightarrow [0,1]$ is a necessity measure on a Boolean algebra $F \subseteq 2^W$ such that for each $\varphi \in \mathcal{L}_D$, the set $\{w \mid e(\varphi, w) = 1\}$ is μ-measurable [78].

The truth evaluation $e(w, \varphi)$ of a DL-formula φ is defined (either 0 or 1) as described in previous Section 3.1. Then e is extended to B-formulae by means of

$G_\Delta(C)$ logic truth-functions and the necessity interpretation of belief, as follows:

- $e(B\varphi, w) = \mu(\{w' \in W \mid e(\varphi, w') = 1\})$, for each crisp φ

- $e(\overline{r}, w) = r$, for all $r \in C$

- $e(\Phi \wedge_G \Psi, w) = \min(e(\Phi, w), e(\Psi, w))$

- $e(\Phi \rightarrow_G \Psi, w) = \begin{cases} 1, & \text{if } e(\Phi, w) \leq e(\Psi, w) \\ e(\Psi, w), & \text{otherwise} \end{cases}$

- $e(\Delta(\varphi), w) = \begin{cases} 1, & \text{if } e(\varphi, w) = 1 \\ 0, & \text{otherwise} \end{cases}$

Then, the truth-degree of a formula Φ in a Necessity Kripke structure M_{BCn} is defined as

$$\|\Phi\|_{M_{BCn}} = \inf_{w \in W} e(\Phi, w)$$

for a $B\varphi$ formula (with φ crisp) the following holds:

$$\|B\varphi\|_{M_{BCn}} = \mu(\{w' \in W \mid e(\phi, w') = 1\})$$

If T is a set of B-formulae, $T \subseteq \mathcal{L}_{BCn}$ one defines the truth-degree of a B-formulae Φ over T as the value

$$\|\Phi\|_T = \inf\{\|\Phi\|_M \mid M \text{ is a } BC_{nec} \text{ model of } T\}$$

Models of BC_{nec} are K structures such that $\|\Phi\|_K = 1$ for each $\Phi \in BC_{nec}$. If $T \subseteq \mathcal{L}_{BCn}$, we will write $T \models_{BC_{nec}} \Phi$ when $\|\Phi\|_K = 1$ for each K model of $BC_{nec} \cup T$.

BC_{nec} Axioms and rules

The axioms and rules for BC_{nec} are set in layers according to the nature of the language $\mathcal{L}_{BCn}$ and the particular uncertainty model chosen here. Namely, we need axioms for the DL-formulae, for the B-formulae and to model the behavior of B-formulae respecting the selected uncertainty model. Then, they are defined in the following way:

1. Axioms and rules of propositional Dynamic logic for DL-formulae (see Section 3.1).

2. Axioms and rules for $G_\Delta(C)$ logic for B-formulae (see Section 3.2).

3. Necessity Axioms (where φ and ψ are DL-formulae)

   (a) $B(\varphi \rightarrow \psi) \rightarrow_G (B\varphi \rightarrow_G B\psi)$

   (b) $B(\varphi \wedge \psi) \equiv_G B\varphi \wedge_G B\psi$

(c) $\neg_G B\bot$

(d) $B\varphi$ for every φ DL-theorem

4. Necessitation rule for B: from φ derive $B\varphi$

Proof within BC_{nec} will be denoted as $\vdash_{BC_{nec}}$.

BC_{nec} Soundness and Completeness

We basically follow the proof schema presented in [47] for Graded Dynamic Deontic logics which, in turn, is based on the proof of Theorem 8.4.9 in [78], with some adaptation. The idea is that a B-modal theory T (consisting of B-formulae) can be represented as a theory T^* over the propositional logic $G_\Delta(C)$.

For each modal formula $B\varphi$ we introduce a propositional variable $p\varphi$. Then, we define a mapping $*$ as follows:

- $(B\varphi)^* = p_\varphi$,
- $(\overline{r})^* = \overline{r}$, for each rational $r \in [0,1]$,
- $(\Phi \wedge_G \Psi)^* = \Phi^* \wedge_G \Psi^*$
- $(\Phi \rightarrow_G \Psi)^* = \Phi^* \rightarrow_G \Psi^*$.
- $(\Delta\Phi)^* = \Delta\Phi^*$

If T is a set of B-formulae, let T^* be the following set of $G_\Delta(C)$ formulae: $T^* = \{\Phi^* \mid \Phi \in T\} \cup \{p_\varphi, \text{for each } \varphi : \vdash_{DL} \varphi\} \cup \{\psi^*, \text{for each Necessity Axiom } \psi\}$.

Lemma 1 *If T is a B-theory and Φ is a B-formula, then*

$$T \vdash_{BC_{nec}} \Phi \quad \textit{iff} \quad T^* \vdash_{G_\Delta(C)} \Phi^*$$

Proof: Assume that $T^* \vdash_{G_\Delta(C)} \Phi$. Let $\alpha_1^*, \ldots, \alpha_n^*$ be a $G_\Delta(C)$-proof of Φ^* in T^*. Then, that sequence can be converted into a BC_{nec}-proof of Φ in T by adding for each formula of the form p_ψ that occurs in $\alpha_1^*, \ldots, \alpha_n^*$, a proof of ψ in PDL and then applying the rule of necessitation for B-formulae. Conversely, assume $T \vdash_{BC_{nec}} \Phi$. Then, a $G_\Delta(C)$-proof of Φ^* in T^* can be obtained by taking the translation of the formulae of one PDL-proof of Φ in T, once the DL-formulae are deleted. Use the fact that every DL-formula provable in a B-theory is a PDL-theorem (see [47]). □

Lemma 2 (Soundness) *For every B-theory T over BC_{nec} and every Φ B-formula,*

$$T \vdash_{BC_{nec}} \Phi \textit{ implies } T \models_{G_\Delta(C)} \Phi$$

Proof: It follows from the fact that the Necessity Axioms are 1-true in every Necessity Kripke structure. □

Theorem 3 (Completeness) *For every B-theory T over BC_{nec} and every B-formula Φ,*

$$T \models_{BC_{nec}} \Phi \textit{ implies } T \vdash_{G_\Delta(C)} \Phi$$

Proof: By Lemma 2 and the Completeness Theorem of the $G_\Delta(C)$ logic it is enough to prove that

$$T \models_{BC_{nec}} \Phi \text{ implies } T^* \models_{G_\Delta(C)} \Phi^*$$

Assume $T^* \not\models_{G_\Delta(C)} \Phi^*$ then, there is a model E of T^*, with evaluation v such that $v(\phi^*) < 1$. We will show that there is a M_v model of T that is not a model of ϕ.

Let $U = \langle W, \rho, e\rangle$ be a Universal model of DL. The Universal models for PDL where introduced in [94] and satisfy the following properties:

- U is a regular Kripke model
- for every formula ϕ in PDL, ϕ is valid in U if and only if ϕ is a theorem of PDL.
- Every regular Kripke model of PDL, can be isomorphically embedded in U.

Let us denote by X_φ the set $\{w \in W \mid e(w, \varphi) = 1\}$ and consider now the following Boolean subalgebra $F \subseteq 2^W$:

$$F = \{X_\varphi : \varphi \text{ is a DL-formula}\}$$

We define a function $\mu : F \to [0, 1]$ as follows: $\mu(X_\varphi) = v(p_\varphi)$, then it can be shown (see [47]) the following items:

(i) μ is well defined and is a necessity measure on F.

To prove that μ is a well defined function is necessary to show that if $X_\varphi = X_\psi$ then $v(p_\varphi) = v(p_\psi)$ and this is a consequence of the second property of the Universal models of DL. On the other hand, to prove that μ is a necessity measure, the different properties corresponding to those measures, i.e. $\mu(\emptyset) = 0$, $\mu(W) = 1$ and $\mu(X_\varphi \cap Y_\psi) = \min(\mu(X_\varphi), \mu(Y_\psi))$, are shown to be satisfied.

(ii) For the structure $M_v = \langle W, \rho, e, \mu\rangle$ and for every B-formulaΦ,

$\|\Phi\|_{M_v} = v(\Phi^*)$.

For this, it is enough to show that for every DL-formula φ, $\|B\varphi\|_{M_v} = v(p_\varphi)$. It is easy to check by induction on the complexity of the B-formula and by definition of μ.

The previous items have just proved that M_v is a necessity Kripke model of T and $\|\Phi\|_{M_v} = v(\Phi^*) < 1$. This concludes the proof. □

5.4 The BC_{prob} Logic

Now, we define a fuzzy modal language over $\mathcal{L}_D$ to reason about the probability of dynamic propositions in a similar way than we defined $\mathcal{L}_{BCn}$. The language of BC_{prob}, $\mathcal{L}_{BCp}$, is built from propositional variables of the form $B\varphi$ for each $\varphi \in \mathcal{L}_{DL}$. Compound formulae are defined in the usual way in the Rational Pavelka logic (RPL) using the Łukasiewicz connectives and truth-constants $\overline{r}$, for each rational $r \in [0,1]$, as follows:

- If $\varphi \in \mathcal{L}_D$ then $\varphi \in \mathcal{L}_{BCp}$.
- If $\varphi \in \mathcal{L}_D$ then $B\varphi \in \mathcal{L}_{BCp}$
- If $r \in \mathbb{Q} \cap [0,1]$ then $\overline{r} \in \mathcal{L}_{BCp}$
- If $\Phi, \Psi \in \mathcal{L}_{BCp}$ then $\Phi \to_L \Psi \in \mathcal{L}_{BCp}$ and $\Phi \& \Psi \in \mathcal{L}_{BCp}$ (where $\&$ and $\to_L$ correspond to the conjunction and implication of Łukasiewicz logic)

Other Łukasiewicz logic connectives for the modal formulae can be defined from $\&$, $\to_L$ and $\overline{0}$: $\neg_L\Phi$ is defined as $\Phi \to_L \overline{0}$, $\Phi \wedge \Psi$ as $\Phi\&(\Phi \to_L \Psi)$, $\Phi \vee \Psi$ as $\neg_L(\neg_L\Phi \wedge \neg_L\Psi)$ and $\Phi \equiv \Psi$ as $(\Phi \to_L \Psi)\&(\Psi \to_L \Phi)$.

Since in this logic a formula $\Phi \to_L \Psi$ is 1-true iff the truth value of Ψ is greater or equal to that of Φ, modal formulae of the type $\overline{r} \to_L B\varphi$ express that the probability of φ is at least r. Formulae of the type $\overline{r} \to_L \Psi$ will be denoted as (Ψ, r). Also we can represent qualitative formulae like $B\varphi \to_L B\psi$ expressing that the probability of ψ is greater or equal than the probability of φ.

In this context, the agent's beliefs will be expressed by a theory $\mathcal{T}$ (a set of B-formulae) containing quantitative and qualitative expressions.

BC_{prop} Semantics

In a similar way as we did for BC_{nec} semantics, we define a BC probabilistic Kripke structure of the following form: $M_{BCp} = \langle W, \{R_\alpha : \alpha \in \Pi\}, e, \mu\rangle$ where $\langle W, \{R_\alpha : \alpha \in \Pi\}, e\rangle$ is regular Kripke model of PDL and $\mu : F \to [0,1]$ is a probabilistic measure on a Boolean algebra $F \subseteq 2^W$ such that for each crisp φ, the set $\{w \mid e(\varphi, w) = 1\}$ is μ-measurable.

The e evaluation is extended as usual to DL-formulae (see previous Section 3.1) and it is extended to B-modal formulae by means of Łukasiewicz logic truth-functions and the probabilistic interpretation of belief, as the following items:

- $e(B\varphi, w) = \mu(\{w' \in W \mid e(\varphi, w') = 1\})$, for each crisp φ

- $e(\overline{r}, w) = r$, for all $r \in Q \cap [0,1]$

- $e(\Phi\&\Psi, w) = \max(e(\Phi, w) + e(\Psi, w) - 1, 0)$

- $e(\Phi \to_L \Psi, w) = \min(1 - e(\Phi, w) + e(\Psi, w), 1)$

Finally, the truth-degree of a $B\varphi$ formula (with φ crisp) in a probabilistic Kripke structure M_{BCp} is as follows

$$\|B\varphi\|_{M_{BCp}} = \mu(\{w' \in W \mid e(\varphi, w') = 1\})$$

Now, given a theory $T \in \mathcal{L}_{BCp}$ (a set of B-formulae) one defines the truth-degree of a B-formulae Φ over T as the value

$$\|\Phi\|_T = \inf\{\|\Phi\|_M \mid M \text{ is a } BC_{prob} \text{ model of } T\}$$

Models of BC_{prob} are M structures such that $\|\Phi\|_M = 1$ for each $\Phi \in BC_{prob}$. If $T \subseteq \mathcal{L}_{BCp}$, we will write $T \models_{BC_{prob}} \Phi$ when $\|\Phi\|_M = 1$ for each M model of $BC_{prob} \cup T$.

BC_{prob} Axioms and rules

In a similar way than in BC_{nec}, the axioms and rules for BC_{prob} are built in layers in the following way:

1. Axioms of propositional Dynamic logic (for DL-formulae), see Section 3.1.
2. Axiom for RPL (for B-formulae), see Section 3.2.
3. Probability Axioms:
   (a) $B(\varphi \to \psi) \to_L (B\varphi \to_L B\psi)$
   (b) $B(\varphi \vee \psi) \equiv_L B\varphi \oplus (B\psi \ominus B(\varphi \wedge \psi))$
   (c) $\neg_L P(\bot)$
   (d) $B\varphi$, for each theorem φ of DL

where $\varphi \oplus \psi$ is a shorthand for $\neg_L\varphi \to_L \psi$ and $\psi \ominus \varphi$ is a shorthand for $\neg_L(\varphi \to_L \psi)$.[1] Deduction rules for BC_{prop} are Modus Ponens (both for $\to$ of DL and for $\to_L$ of RPL) and Necessitation for the modality B.

Proof within BC_{prob} will be denoted as $\vdash_{BC_{prob}}$. Given a set of formulae $T \cup \Phi \subset \mathcal{L}_{BCp}$, we define a provability degree of ϕ over T as

$$|\Phi|_T = \sup\{r \mid T \vdash_{BC_{prob}} \overline{r} \to_L \Phi\}$$

[1]Note that in Łukasiewicz logic $(x \Rightarrow_L 0) \Rightarrow_L y = \min(1, x + y)$ and $(x \Rightarrow_L y) \Rightarrow_L 0 = max(0, x - y)$

BC_{prob} Soundness and Completeness

It is easy to check that the above defined BC_{prob} is sound with respect to the class of BC probabilistic Kripke structures.

Lemma 4 (Soundness) *For every B-theory T over BC_{prob} and every Φ B-formula,*

$$T \vdash_{BC_{prob}} \Phi \text{ implies } T \models_{RPL} \Phi$$

Proof: It follows from the fact that the Probability Axioms are 1-true in every Probability Kripke structure (cf. [78, Lemma 8.4.5]). □

Although we cannot apply the same technique used for BC_{nec} to get a completeness theorem like Theorem 3 since RPL is not strong complete for arbitrary theories, it can be proved that the Pavelka-style completeness of RPL (see Section 3.2) extends to BC_{prob} by easily adapting the proof of Theorem 8.4.9 in [78] for the logic $FP(RPL)$, a probability logic over classical propositional logic, to our case of a probability logic over PDL, making use of the of notion universal model for PDL. We therefore omit the proof.

Theorem 5 (Pavelka completeness of BC_{prob}) *For every B-theory T over BC_{prob} and every ϕ B-formula, we have the following equality:*

$$\|\Phi\|_T = |\Phi|_T.$$

Notice that the above completeness result implies in particular that $T \models_{BC_{prob}} \Phi$ iff $\sup\{r \mid T \vdash_{BC_{prob}} \overline{r} \rightarrow_L \Phi\} = 1$.

5.5 Conclusions

We have presented two formalizations for the BC context following a schema based on a fuzzy-modal approach. In this way, we have shown that different uncertainty models may be represented using an appropriate many-valued logic and a set of axioms according to the chosen uncertainty model. The fuzzy modal approach used for belief representation provides us a solid logical background including soundness and completeness results. Preliminary approaches to the belief context in the g-BDI agent model can be seen in [29, 30, 31].

Comparing our BC logics BC_{nec} and BC_{prob} to Rao and Georgeff's BDI logic, two comments are in order. First of all, our approach allows us to reason about (two different notions of) *graded* beliefs, while the classical approach does not. A second comment, abstracting from the gradedness of our logics, is about comparing the axioms for the chosen uncertainty models, respectively necessity and probability axioms the axiomatics presented for BC_{nec} and BC_{prob} with the one proposed by Rao and Georgeff for the beliefs in his BDI logic, i.e., KD45 axioms [123]. In both

proposals for BC axioms we are also including K and D axioms, but axioms related to introspection (i.e., axioms 4 and 5) are not considered because the the definition of BC language does not allow nested modalities. Even though formulae with the nested modalities e.g, $B(B\varphi)$, are not allowed in our BC language because $B\varphi$ is a fuzzy formula and we only apply the B-modality over two-valued (crisp) ones, we could partially overcome this shortcoming by using for instance the Δ operator. This approach would become indeed valuable in a multi-agent framework where an agent needs to represent the beliefs she has about other agents and is described in Chapter 8.

In the next Chapter we describe the other two mental contexts of the g-BDI agent model related to the agent preferences. The first one, the Desire context (DC) is in charge of dealing with the ideal agent preferences, both positive and negative. The second one, the intention context (IC) represents the agent intentions which are the desires the agent decides to follow after analyzing the cost/benefit relation associated with a plan of actions to achieve them.

All you have to decide is what to do with the time that has been given to you.

J.R.R. Tolkien

Chapter 6

Desire and Intention Contexts

6.1 Introduction

In this chapter we present the contexts of the g-BDI agent model in charge of dealing with the agent's preferences: the Desire and Intention contexts. As we have mentioned in Section 2.4 preferences are essential for making intelligent choices in complex situations, for mastering large sets of alternatives, and for coordinating a multitude of decisions. In particular, preferences are the proactive attitudes in intentional agents. From these positive preferences or desires the agent may choose which ones it will intend to achieve through a suitable plan of actions. Negative preferences are also considered in modelling different AI problems and particularly in multiagent systems. For an intentional agent negative preferences may represent restrictions or rejections over the possible worlds it can reach.

In next Section 6.2 the Desire context (DC) is introduced, to represent the agent's desires. Desires represent the *ideal* agent's preferences regardless of the agent's current perception of the environment and regardless of the cost involved in actually achieving them. We deem important to distinguish what is positively desired from what is not rejected. According to the works on bipolarity representation of preferences by Benferhat et.al. [11], described in Section 2.4, positive and negative information may be modeled in the framework of possibilistic logic. Inspired by this work, we suggest to formalize also positive and negative desires. Positive desires represent what the agent would like to be the case. Negative desires correspond to what the agent rejects or does not want to occur. Furthermore, positive and negative desires can be graded to represent different levels of preference or rejection, respectively. When dealing with both kinds of preferences it is also natural to express indifference, meaning that we have neither a positive nor a negative preference over an object. To represent and reason about the agent bipolar preferences in the DC the definition of its logical components (i.e. the language, axioms and inference rules) is needed.

Firstly, we define a language to express positive and negative desires. Secondly, using an appropriate axiomatic the behavior of these preferences is modelled. In a

similarly way than in the BC context, a modal many-valued approach is used to deal with the desire degrees and a layered structure of axioms is set. As for combining one kind of desires (positive or negative) usually the conjunction of positive (resp. negative) preferences should produce a higher positive (resp. negative) preference. The disjunction degree of positive (resp. negative) preferences is computed as the minimum of the desire degrees, following the intuition that if the disjunction is satisfied unless the minimum of satisfaction (rejection) is guaranteed. Then, the axiomatic and the inference rules are defined to capture these combination properties for positive and negative desires independently. The language with this set of axioms constitute the basic logic framework for the Desire context (DC schema).

As desire are ideal preferences, we consider that it may be somewhat controversial and domain dependent to set (normative) general restrictions about e.g. positive (negative) desires both on some formula and its negation, or also between the positive and negative desires on a same given formula. In this direction, we present the basic framework DC for the bipolar desires representation without including additional restrictions. Then, besides the basic framework some alternative constraints are analyzed in this work, resulting in different logical schemas or theories. Some of the possible constraints are commented below.

Following the intuition (that seems valid in most domains) that a formula and its negation cannot be both desired (respectively rejected) at the same time, some restrictions over the desires (positive and negative) respect a formula may be established. Since positive and negative preferences are stated separately, it is worthwhile to consider whether any consistency condition may be imposed between them. Benferhat et al. [12] present a *coherence* condition restricting what is desirable to what is tolerated. Namely, any world satisfying at least one positive desire should also satisfy all the constraints induced by the negative preferences. We have proposed a more restrictive approach in [30] imposing the condition that if a world is rejected to some extent, it cannot be desired at the same time, by an agent. And conversely, if a solution is somewhat desired it cannot be rejected. This is a strong restriction not allowing to represent that a world may be partially desired because of some aspects and partially rejected because of others, it may be useful to represent preferences in particular problems.

To formalize these alternative preference models, a Basic Schema for the Desire Context (DC) is first presented and afterward, we refine it by adding the different preference properties above mentioned. For each consistency schema, the corresponding semantics and axioms are presented.

Later on, in Section 6.3, the Intention context (IC) to represent the agent's intentions, is described. We follow the model introduced by Rao and Georgeff [123, 125], in which an intention is considered a fundamental pro-attitude with an explicit representation. However, as in the work of Cohen and Levesque [41], in our approach, intentions result from the agent's beliefs and desires and then, we do not consider them as a basic attitude. Intentions, as well as desires, represent the agent prefer-

ences. However, we consider that intentions cannot depend just on the benefit, or satisfaction, of reaching a desire φ —represented in $D^+\varphi$, but also on the world's state w and the cost of transforming it into a world w_i where the formula φ is true. By allowing degrees in intentions we represent a measure of the cost/benefit relation involved in the agent's actions toward the desired goal. A similar semantics for intentions is used in [142], where the net value of an intention is defined as the difference between the value of the intention outcome and the cost of the intention. In [124], this relation is resumed in the payoff function over the different paths. The formalization of the intention semantics is difficult, because it does not depend only on the formula intended, but also on the plan that the agent executes to achieve a state where the formula is valid. Our work evolved in this aspect as can be seen in [29], [30] and [31].

In our model, the positive and negative desires are used as pro-active and restrictive tools respectively in order to set intentions. Note that intentions depend on the agent's knowledge about the world, which may allow —or not— the agent to choose a plan to change the world into a desired one.

We represent in this context two kinds of graded intentions, intention of a formula φ considering the execution of a particularly plan α, noted $I_\alpha\varphi$, and the final intention to φ, noted $I\varphi$, which takes into account the best path to reach φ. As in the other contexts, if the degree of $I\varphi$ is δ, it may be considered that the truth degree of the expression "φ is intended" is δ. The intention to make φ true must be the consequence of finding a feasible plan α, that permits to achieve a state of the world where φ holds.

6.2 Desire Context (DC)

6.2.1 DC Language

To represent positive and negative desires over formulae of a basic propositional language $\mathcal{L}$ we introduce in such a language two modalities. Thus, the theory associated to the Desire Context will consist of a set of modal formulae from the expanded language $\mathcal{L}_{DC}$ representing all the available information about the agent's desires. It can be the case that the context theory needs only a given subset of the modal language, for instance, when the agent's desires are only expressed over literals or conjunctions of them.

The language $\mathcal{L}_{DC}$ is defined over a classical propositional language $\mathcal{L}$ (built from a countable set of propositional variables *Var* with connectives $\wedge$, $\rightarrow$ and $\neg$) expanded with two (fuzzy) modal operators D^+ and D^-. $D^+\varphi$ reads as "φ is positively desired" and its truth degree represents the agent's level of satisfaction would φ become true. $D^-\varphi$ reads as "φ is negatively desired" (or "φ is rejected") and its truth degree represents the agent's level of disgust on φ becoming true. As in the BC logic, we will use a modal many-valued logic to formalize graded desires. We use again Rational Pavelka logic (i.e. Łukasiewicz logic expanded with rational

truth-constants) as the base logic.

More precisely, formulae of the expanded language $\mathcal{L}_{DC}$ are defined as follows:

- If $\varphi \in \mathcal{L}$ then $\varphi \in \mathcal{L}_{DC}$
- If $\varphi \in Sat(\mathcal{L})$[1] then $D^-\varphi, D^+\varphi \in \mathcal{L}_{DC}$
- If $r \in Q \cap [0,1]$ then $\overline{r} \in \mathcal{L}_{DC}$
- If $\Phi, \Psi \in \mathcal{L}_{DC}$ then $\Phi \to_L \Psi \in \mathcal{L}_{DC}$ and $\neg_L \Phi \in \mathcal{L}_{DC}$ (other Łukasiewicz logic connectives, like $\wedge_L, \vee_L, \equiv_L$ are definable from $\neg_L$ and $\to_L$)

We will call a modal formula *closed* when every propositional variable is in the scope of a D^+ or a D^- operator.
As in $\mathcal{L}_{BC}$, the notation $(D^+\psi, r)$, with $r \in [0,1] \cap \mathbb{Q}$, will be used as a shortcut of $\overline{r} \to_L D^+\psi$, and reads as: the level of positive desire of ψ is at least r. Analogously for $(D^-\psi, r)$ and $\overline{r} \to_L D^-\psi$.

In this context, the agent's preferences will be expressed by a theory $\mathcal{T}$ (a set of $\mathcal{L}_{DC}$-formulae) containing quantitative expressions about positive and negative preferences, like $(D^+\varphi, \alpha)$ or $(D^-\psi, \beta)$, as well as qualitative expressions like $D^+\psi \to_L D^+\varphi$ (resp. $D^-\psi \to_L D^-\varphi$), expressing that φ is at least as preferred (resp. rejected) as ψ. In particular $(D^+\phi_i, 1) \in \mathcal{T}$ means that the agent has maximum preference in ϕ_i and is fully satisfied if it is true. While $(D^+\phi_j, \alpha) \notin \mathcal{T}$ for any $\alpha > 0$ means that the agent is indifferent to ϕ_j and the agent does not benefit from truth of ϕ_j becoming true. Analogously, $(D^-\psi_i, 1) \in \mathcal{T}$ means that the agent absolutely rejects ϕ_i and thus the states where ψ_i is true are totally unacceptable. If $(D^-\psi_j, \beta) \notin \mathcal{T}$ for any $\beta > 0$ it simply means that ψ_j is not rejected.

6.2.2 Semantics for DC

Many people can argue that considering the desires as a proactive attitude, then, reasoning about desires on disjunctions of formulae may have no sense. In most cases we may have plans for achieving φ or ψ individually, or for both $(\varphi \wedge \psi)$ but not for achieving non-deterministically $\varphi \vee \psi$. But since we define the basic language as a propositional language it is necessary to define the semantics in terms of preferences for disjunctive formulae, and leave the selection of what formulae should been considered to reason about desires to the definition of a particular theory.

According to the semantics presented in [11], the degree of positive desire for (or level of satisfaction with) a disjunction of desires $\varphi \vee \psi$ is taken to be the minimum of the degrees for φ and ψ. Intuitively, if an agent desires $\varphi \vee \psi$ then it is ready to

[1] Sat($\mathcal{L}$) represent the set of satisfiable formulae of $\mathcal{L}$ and thus, excluding to have positive and negative desires on a contradiction ($\bot \notin Sat(\mathcal{L})$).

accept the situation where the less desired goal becomes true, and hence to accept the minimum satisfaction level produced by one of the two desires. In contrast the satisfaction degree of reaching both φ and ϕ can be strictly greater than reaching one of them separately. These are basically the properties of the *guaranteed possibility* measures (see e.g. [10]). Analogously, we assume the same model for the degrees of negative desire or rejection, that is, the rejection degree of $\varphi \vee \psi$ is taken to be the minimum of the degrees of rejection for φ and for ψ separately, while nothing prevents the rejection level of $\varphi \wedge \psi$ be greater than both.

The intended DC models are Kripke structures $M = \langle W, e, \pi^+, \pi^- \rangle$ where W and e are defined as in the BC semantics and π^+ and π^- are preference distributions over worlds, which are used to give semantics to positive and negative desires:

- $\pi^+ : W \to [0,1]$ is a distribution of positive preferences over the possible worlds. In this context $\pi^+(w) < \pi^+(w')$ means that w' is more preferred than w.
- $\pi^- : W \to [0,1]$ is a distribution of negative preferences over the possible worlds: $\pi^-(w) < \pi^-(w')$ means that w' is more rejected than w.

The truth evaluation for non-modal formulae $e : \mathcal{L} \times W \to \{0,1\}$ is defined in the usual (classical) way. It is extended to atomic modal formulae $D^-\varphi$ and $D^+\varphi$ by:

- $e(D^+\varphi, w) = \inf\{\pi^+(w') \mid e(\varphi, w') = 1\}$
- $e(D^-\varphi, w) = \inf\{\pi^-(w') \mid e(\varphi, w') = 1\}$

together with the assumption that $\inf \emptyset = 1$. This is extended to compound modal formulae by means of the usual truth-functions for Łukasiewicz connectives. Notice that the evaluation $e(w, \Phi)$ of a modal formula Φ only depends on the formula itself —represented in the preference measure over the worlds where the formula is true— and not on the actual world $w \in W$ where the agent is situated. In such a case, we will also write $e_M(\Phi)$ for $e(w, \Phi)$. This is consistent with the intuition that desires represent ideal preferences of an agent, regardless of the actual world and regardless of the cost of moving to a world where the desire is satisfied.

We will write $M \models \Phi$ when $e(\Phi, w) = 1$ for all $w \in W$. Moreover, let $\mathcal{M}_{\mathcal{DC}}$ be the class of all Kripke structures $M = \langle W, e, \pi^+, \pi^- \rangle$. Then, for each subclass of models $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{DC}}$, given a theory T and a formula Φ, we will write $T \models_{\mathcal{M}} \Phi$ if $M \models \Phi$ for each model $M \in \mathcal{M}$ such that $M \models \Psi$ for all $\Psi \in T$.

6.2.3 DC Axioms and Rules

To axiomatize the logic with above intended preference-based semantics we need to combine classical logic axioms for non-modal formulae with Rational Pavelka logic

axioms for modal formulae. Also, additional axioms characterizing the behavior of the modal operators D^+ and D^- are needed. As already mentioned, a conjunctive combination of one kind of desires (either positive or negative) preferences may be attached a strictly higher preference value, while the preference value of a disjunctive combination of either positive or negative desires is taken as the minimum of the desire degrees, following the intuition that at least the minimum of satisfaction (rejection) is guaranteed. The following axioms and inference rules aim at capturing these combination properties, considering positive or negative desires independently. We define the basic set of axioms and rules for the DC logic as follows:
Axioms:

(CPC) Axioms of classical logic for non-modal formulae

(RPL) Axioms of Rational Pavelka logic for modal formulae

(DC0$^+$) $D^+(A \vee B) \equiv_L D^+A \wedge_L D^+B$

(DC0$^-$) $D^-(A \vee B) \equiv_L D^-A \wedge_L D^-B$

And the Inference Rules as:

(MP1) modus ponens for $\rightarrow$

(MP2) modus ponens for $\rightarrow_L$

Introduction of D^+ and D^- for implications:

(ID$^+$) from $\varphi \rightarrow \psi$ derive $D^+\psi \rightarrow_L D^+\varphi$

(ID$^-$) from $\varphi \rightarrow \psi$ derive $D^-\psi \rightarrow_L D^-\varphi$.

The notion of proof, denoted $\vdash_{DC}$, is defined as usual from the above axioms and inference rules.

Notice that the two axioms (DC0$^+$) and (DC0$^-$) define the behavior of D^- and D^+ with respect to disjunctions.

The formalization we present for D^- is somewhat different from the approach presented by Benferhat et al. in [11], where they used a necessity function i.e., considering $D^-\phi$ as $N(\neg\phi)$. But in their approach the axiomatic is equivalent since the axiom (DC0$^-$) we present results from the necessity axiom i.e., $N(A \wedge B) \equiv N(A) \wedge_L N(B)$.

Finally, the introduction rules for D^+ and D^- state that the degree of desire is monotonically decreasing with respect to logical implication. Moreover, an easy consequence that these rules allow is that equivalent desire degrees are preserved by Boolean equivalence.

Lemma 6 *If $\vdash$ denotes deduction in classical propositional calculus, then $\vdash \varphi \equiv \psi$ implies $\vdash_{DC} D^+\varphi \equiv_L D^+\psi$ and $\vdash_{DC} D^-\varphi \equiv_L D^-\psi$.*

The above axiomatization is correct with respect to the defined semantics.

Lemma 7 (soundness) *Let T be a theory and Φ a formula. Then $T \models_{\mathcal{M}_{\mathcal{DC}}} \Phi$ if $T \vdash_{DC} \Phi$.*

Proof: It is a matter of routine to check that the axioms are valid in each DC-model and that the inference rules preserve validity in each DC-model. □

Moreover, the basic DC logic is complete as well for finite theories of closed (modal) formulae.

Theorem 8 (completeness) *Let T be a finite theory of closed formulae and Φ a closed formula. Then $T \models_{\mathcal{M}_{\mathcal{DC}}} \Phi$ iff $T \vdash_{DC} \Phi$.*

Proof: We basically follow the proof of Theorem 8.4.9 in [78], with some adaptations.

Assume $p_1, \ldots, p_n$ contain at least all the propositional variables involved in T and Φ, and let $Nor = \{\chi_i\}_{i=1,2^{2^n}}$ the set of 2^{2^n} non logically equivalent Boolean formulae in DNF built from the p_i's. For each non-modal φ built from the p_i's, let $\varphi_{NF} \in Nor$ denote its corresponding normal form. Then for each modal Φ let us denote by Φ_{NF} the result of replacing each atomic modal component of the form $D^+\varphi$ or $D^-\varphi$ by $D^+\varphi_{NF}$ or $D^-\varphi_{NF}$ respectively. Finally, for each modal theory S let us denote by S_{NF} the result of replacing each $\Phi \in S$ by Φ_{NF}.

The idea is that the modal theory T can be represented as a (finite) theory over the propositional logic RPL. For each modal formula $D^+\varphi$ introduce a propositional variable p^+_ϕ, and for each $D^-\varphi$ another propositional variable p^-_φ. Then define a mapping $*$ as follows:

- $(D^+\varphi)^* = p^+_\varphi$,
- $(D^-\varphi)^* = p^-_\varphi$,
- $(\overline{r})^* = \overline{r}$, for each rational $r \in [0,1]$,
- $(\Phi \&_L \Psi)^* = \Phi^* \&_L \Psi^*$
- $(\Phi \rightarrow_L \Psi)^* = \Phi^* \rightarrow_L \Psi^*$.

If S is a set of modal formulae, let $S^* = \{\Phi^* \mid \Phi \in S\}$.

Now, let $\mathcal{DC} = \{\Phi \mid \Phi$ is an instance of modal axioms (DC3$^+$) and (DC3$^-$)$\} \cup \{D^+\varphi \rightarrow_L D^+\psi, D^-\varphi \rightarrow_L D^-\psi \mid \varphi \rightarrow \psi$ theorem of CPC $\}$. We next show that the following statements are equivalent:

1) $T \models_{DC} \Phi$
2) $T^* \cup \mathcal{DC}^* \models_{RPL} \Phi^*$
3) $T^*_{NF} \cup (\mathcal{DC}_{NF})^* \models_{RPL} \Phi^*_{NF}$
4) $T^*_{NF} \cup (\mathcal{DC}_{NF})^* \vdash_{RPL} \Phi^*_{NF}$
5) $T^* \cup \mathcal{DC}^* \vdash_{RPL} \Phi^*$
6) $T \vdash_{DC} \Phi$

$1 \Rightarrow 2$: Let us assume $T^* \cup \mathcal{DC}^* \not\models_{RPL} \Phi^*$. This means there is an RPL-evaluation v model of $T^* \cup \mathcal{DC}^*$ and $v(\Phi^*) < 1$. We build then a model $M_v = \langle W, e, \pi^+, \pi^- \rangle$ as follows:

- W is the set of Boolean evaluations of the propositional variables $q_1, ..., q_n$;
- $e(w, q) = w(q)$, for each propositional variable q, and $e(w, \cdot)$ is extended to Boolean formulae as usual;
- $e(w, \overline{r}) = r$ for each rational $r \in [0, 1]$;
- $e(w, D^+\varphi) = v(p^+_\varphi)$ and $e(w, D^-\varphi) = v(p^-_\varphi)$, and $e(w, \cdot)$ is extended to compound modal formulae using RPL connectives;
- $\pi^+(w) = v(p^+_{A_w})$ and $\pi^-(w) = v(p^-_{A_w})$, where A_w is the elementary conjunction built with literals from the propositional variables $q_1, \ldots, q_n$ such that $e(w, A_w) = 1$ and $e(w', A_w) = 0$ if $w' \neq w$;

Since $M_v \models \varphi \equiv \vee_{w \in W} A_w$, it is easy to check that, $e(w, D^+\varphi) = \inf\{\pi^+(w') \mid e(w', \varphi) = 1\}$ and $e(w', D^-\varphi) = \inf\{\pi^-(w') \mid e(w', \varphi) = 1\}$. Therefore M_v is DC model, and since by construction $e(w, \Psi) = v(\Psi^*)$ for all modal formula Ψ and worlds $w \in W$, we also have in particular $e(w, \Psi) = v(\Psi^*) = 1$ for all $\Psi \in T^*$ and $e(w, \Phi) = v(\Phi^*) < 1$ and hence $T \not\models_{DC} \Phi$.

$2 \Rightarrow 3$: Assume e is a RPL-evaluation of the propositional variables $p_{\varphi_{NF}}$ which is a model of $T^*_{NF} \cup (\mathcal{DC}_{NF})^*$ but $e(\Phi^*_{NF}) < 1$. Then, extend e to propositional variables p^+_φ and p^-_φ by putting $e'(p^+_\varphi) = e(p^+_{\varphi_{NF}})$ and $e'(p^-_\varphi) = e(p^-_{\varphi_{NF}})$. It is easy to check that e' is such that $e'(\Phi^*) = e((\Phi_{NF})^*)$ for any modal formula Φ, and hence e' is a model of $T^* \cup \mathcal{DC}^*$ and $e'(\Phi^*) = e(\Phi^*_{NF}) < 1$.

$3 \Rightarrow 4$: Since $T^*_{NF} \cup (\mathcal{DC}_{NF})^*$ is a finite theory (recall that there are finitely-many formulae in Nor), then 4) follows from 3) by the finite strong standard completeness of RPL.

$4 \Rightarrow 5$: Using Lemma 6, if $\vdash \varphi \equiv \psi$ (in classical propositional logic) then $T^* \cup \mathcal{DC}^*$ proves in RPL both $p^+_\varphi \equiv_L p^+_\psi$ and $p^-_\varphi \equiv_L p^-_\psi$, and hence $T^* \cup \mathcal{DC}^* \vdash_{RPL} \Phi^* \equiv_L (\Phi_{NF})^*$ for each modal formula Φ.

$5 \Rightarrow 6$: Let $\Psi^*_1, \ldots, \Psi^*_n$ be a DC-proof of Φ^* from $T^* \cup \mathcal{DC}^*$. This is converted into a DC-proof of Φ from T by adding for each Φ^*_i which is of the form $p^+_\varphi \rightarrow_L p^+_\psi$ (resp. $p^-_\varphi \rightarrow_L p^-_\psi$) with $\varphi \rightarrow \psi$ being a theorem of CPC, a proof of $\varphi \rightarrow \psi$ in CPC and then applying the rule of introduction of D^+ (resp. D^-) for implications.

$6 \Rightarrow 1$: This is soundness.

□

Example 1 *María, who lives in busy Buenos Aires, wants to relax for a few days in an Argentinian beautiful destination. She activates a personal agent, based on our DC logical framework, to get an adequate plan, i.e. a tourist package, that satisfies her preferences. She would be very happy going to a mountain place (m), and rather happy practicing rafting (r). In case of going to a mountain place she would like to go climbing (c). On top of this, she wouldn't like to go farther than 1000km from Buenos Aires (f). She is stressed and would like to get to the destination with a short trip. The user interface that helps her express these desires ends up generating a desire theory as follows:*

$\mathcal{T}_{\mathcal{D}} = \{(D^+m, 0.8), (D^+r, 0.6), D^+m \rightarrow_L D^+c, (D^-f, 0.7)\}$

Once this initial desire theory is generated the tourist advisor personal agent deduces a number of new desires:

$\mathcal{T}_{\mathcal{D}} \vdash_{DC} (D^+(m \wedge r), 0.8)$,
$\mathcal{T}_{\mathcal{D}} \vdash_{DC} (D^+(m \vee r), 0.6)$,
$\mathcal{T}_{\mathcal{D}} \vdash_{DC} (D^+c, 0.8)$

As María would indeed prefer much more to be in a mountain place doing rafting she also expresses the combined desire with a particularly high value: $(D^+(m \wedge r), 0.95)$. *Notice that the extended theory*

$$\mathcal{T}'_{\mathcal{D}} = \mathcal{T}_{\mathcal{D}} \cup \{(D^+(m \wedge r), 0.95)\}$$

remains consistent within DC.

The basic logical schema *DC* puts almost no constraint on the strengths for the positive and negative desires of a formula φ and its negation $\neg\varphi$. This is in accordance with considering desires as ideal preferences and hence it may be possible for an agent to have contradictory desires. Indeed, the only indirect constraint *DC* imposes is the following one: if a theory T derives $(D^+\varphi, r)$ and $(D^+\neg\varphi, s)$ then, due to axiom $(DC0^+)$ and rule (ID^+), T also derives both $(D^+\psi, \min(r, s))$ and $(D^+\neg\psi, \min(r, s))$ for any ψ.

In the following section, different properties are added to the preferences as to represent some constraints between the positive and negative desires of a formula and its negation.

6.2.4 Consistency Schemas

The basic schema for preference representation and reasoning provided by the DC logic may be felt too general for some classes of problems and we may want to restrict the allowed assignment of degrees of positive and negative desire for a formula φ and for its negation $\neg\varphi$. For instance, in the case of considering positive desires

as proactive attitudes, it is not an efficient approach to allow to assign non-zero degrees to $D^+\varphi$ and to $D^+\neg\varphi$, since the agent will be looking for plans toward opposite directions, some plans leading to satisfy φ and some others to satisfy $\neg\varphi$.

In the following subsections three different extensions or schemas are proposed to show how different consistency constraints between positive and negative desires can be added to the basic logic, both at the semantical and syntactical level. These different schemas allow us to define different types of agents. Each agent type will accept (respectively restrict) desire formulae in its theory depending on its defined constraints according to the chosen schema.

DC_1 Schema

It may be natural in some domain applications to forbid to simultaneously have positive (in the sense of > 0) desire degrees for $D^+\varphi$ and $D^+\neg\varphi$. This constraint and the corresponding one for negative desires amounts to require that the following additional properties for the truth-evaluations be satisfied in the intended models:

- $\min(e(D^+\varphi, w), e(D^+\neg\varphi, w)) = 0$, and
- $\min(e(D^-\varphi, w), e(D^-\neg\varphi, w)) = 0$.

At the level of Kripke structures, this corresponds to require some extra conditions over π^+ and π^-, namely:

- $\inf_{w\in W} \pi^+(w) = 0$ and
- $\inf_{w\in W} \pi^-(w) = 0$

These conditions are a kind of *anti-normalization* conditions for π^+ and π^-, in the sense that they require the existence of at least one world that is not desired and one world that is not rejected. Let $\mathcal{M}_{\mathcal{DC}1}$ denote the subclass of models satisfying these conditions.

For instance, following this schema an agent's theory $\mathcal{T}_D$ should not simultaneously contain the formulae $(D^+m, 0.8)$ and $(D^+(\neg m), 0.4)$, or the formulae $(D^-f, 0.7)$ and $(D^-(\neg f), 0.5)$.

At the syntactic level these conditions are equivalent to add to the basic axiomatic for DC the following two axioms:

(DC1$^+$) $D^+\varphi \wedge_L D^+(\neg\varphi) \rightarrow_L \bar{0}$ (or equivalently $D^+(\top) \equiv_L \bar{0}$)

(DC1$^-$) $D^-\varphi \wedge_L D^-(\neg\varphi) \rightarrow_L \bar{0}$ (or equivalently $D^-(\top) \equiv_L \bar{0}$)

We will denote by DC_1 the extension of DC system with the above two axioms (DC1$^+$) and (DC1$^+$), and by $\vdash_{DC_1}$ the corresponding notion of proof.

Theorem 9 (completeness) *Let T be a finite modal theory of closed formulae and Φ a closed formula. Then $T \models_{\mathcal{M}_{\mathcal{DC}1}} \Phi$ iff $T \vdash_{DC_1} \Phi$.*

Proof: The proof runs like in Theorem 8 by adding to the $\mathcal{DC}$ theory the instances of axioms (DC1$^+$) and (DC1$^-$) and with the obvious modifications. □

DC_2 Schema

The above logical schema DC_1 does not put any restriction on positive and negative desires for the same goal (any classically satisfiable formula). According to Benferhat et al. in [12], a coherence condition between positive and negative desires should be considered, namely, an agent cannot desire to be in a world more than the level at which it is tolerated (not rejected). This condition, translated to our framework, amounts to require in the Kripke structures the following constraint between the preference distributions π^+ and π^-:

- $\forall w \in W,\ \pi^+(w) \leq 1 - \pi^-(w)$

To formulate the corresponding axiomatic counterpart that faithfully accounts for the above condition, we consider $\mathcal{M}_{DC_2}$ the subclass of DC-Kripke structures $M = (W, e, \pi^+, \pi^-)$ satisfying the above constraint between $\pi^+$and π^-. Note that $\pi^+(w) \leq 1 - \pi^-(w)$ iff $\pi^+(w) \otimes \pi^-(w) = 0$. [2]

To capture at the syntactical level this class of structures, we consider the extension of the DC system with the following axiom:

(DC2) $(D^+\varphi \otimes D^-\varphi) \rightarrow_L \bar{0}$

We will denote by DC_2 the extension of DC with the axiom (DC2).[3]

Notice that this axiom is valid in every DC-structure $M = (W, e, \pi^+, \pi^-) \in \mathcal{M}_{DC_2}$. Indeed, for any non modal φ, we have:

$e_M(D^+\varphi \otimes D^-\varphi) =$
$\inf\{\pi^+(w) \mid e(w, \varphi) = 1\} \otimes \inf\{\pi^-(w) \mid e(w, \varphi) = 1\} =$
$\inf\{\pi^+(w) \otimes \pi^-(w') \mid e(w, \varphi) = e(w', \varphi) = 1\} \leq$
$\inf\{\pi^+(w) \otimes \pi^-(w) \mid e(w, \varphi) = 1\} = 0.$

that is, for any φ, the evaluations of $D^+\varphi$ and $D^-\varphi$ are such that $e_M(D^+\varphi) \leq 1 - e_M(D^-\varphi)$.

Conversely, if the (DC_2) axiom is valid in a DC-structure $M = (W, e, \pi^+, \pi^-)$ then it must necessarily satisfy the condition $\pi^+(w) \leq 1 - \pi^-(w)$ for any $w \in W$, i.e. $M \in \mathcal{M}_{DC_2}$.

Proof: W.l.o.g., we can assume that W is such that $e(w, \cdot) = e(w', \cdot)$ iff $w = w'$. Then, for each $w \in W$ consider the formula $\varphi_w = (\bigwedge_{p_i \in l^+} p_i) \wedge (\bigwedge_{p_i \in l^-} \neg p_i)$, where $l^+ = \{p \in Var \mid e(w, p) = 1\}$ and $l^- = \{p \in Var \mid e(w, p) = 0\}$. It is clear that $e(v, \varphi_w) = 1$ iff $v = w$, and hence $e(w, D^+\varphi_w) = \pi^+(w)$ and $e(w, D^-\varphi_w) = \pi^-(w)$. Therefore, $e_M(D^+\varphi_w \otimes D^-\varphi_w) = 0$ iff $\pi^+(w) \otimes \pi^-(w) = 0$. □

[2] Here we use the same symbol as the Łukasiewicz connective $\otimes$ to denote its corresponding truth-function on $[0, 1]$, i.e. $x \otimes y = \max(x + y - 1, 0)$ for any $x, y \in [0, 1]$.

[3] An equivalent presentation of axiom (DC_2) is $D^+\varphi \rightarrow_L \neg_L D^-\varphi$.

These properties, together with a suitable adaptation of the proof of Theorem 8, lead to the following completeness result for DC_2.

Theorem 10 (completeness) *Let T be a finite theory of closed formulae and Φ a closed formula. . Then $T \models_{\mathcal{M}_{\mathcal{DC}2}} \Phi$ iff $T \vdash_{DC_2} \Phi$.*

DC_3 Schema

Stronger consistency condition between positive and negative preferences was considered in [30], requiring that if a world is rejected to some extent, it cannot be positively desired at all. And conversely, if a goal (any satisfiable formula) is somewhat desired it cannot be rejected. Indeed, at the semantical level, this amounts to require the intended DC-models $M = (W, e, \pi^+, \pi^-)$ to satisfy the following condition for any $w \in W$:

- $\pi^-(w) > 0$ implies $\pi^+(w) = 0$
  (or equivalently, $\min(\pi^+(w), \pi^-(w)) = 0$)

This is a stronger condition than the one presented in DC_2 schema and may be suitable to represent preferences for some particular domains. We will denote by $\mathcal{M}_{DC_3}$ the subclass of DC-Kripke structures satisfying this latter condition.

At the syntactic level, the corresponding axiom that faithfully represents this consistency condition is the following one:

(DC3) $(D^+\varphi \ \wedge_L D^-\varphi) \rightarrow_L \bar{0}$

We will denote by DC_3 the extension of the DC system by the above axiom (DC3), and by $\vdash_{DC_3}$ the corresponding notion of proof.

Theorem 11 (completeness) *Let T be a finite theory of closed formulae and Φ a closed formula. Then $T \models_{\mathcal{M}_{\mathcal{DC}3}} \Phi$ iff $T \vdash_{DC_3} \Phi$.*

Proof: Again it is an easy adaptation of the proof of Theorem 8. □

Example 2 *(Example 1 continued)*

María, a few days later, breaks her ankle. She activates the recommender agent to reject the possibility of going climbing (c). If María selects for the agent the schema DC_1, the agent simply adds the formula $(D^-c, 1)$ into the former desire theory $\mathcal{T}'_{\mathcal{D}}$, yielding the new theory

$$\mathcal{T}''_{\mathcal{D}} = \{(D^+m, 0.8), (D^+r, 0.6), (D^+(m \wedge r), 0.95), (D^+c, 0.85), (D^-f, 0.7), (D^-c, 1)\} \ ,$$

as the schema allows for opposite desires.[4]

[4]The fact of having both positive and negative desires may be handled in different ways depending on the kind of agent behaviour. For instance, if the agent follows [11]'s approach, where negative desires are used as strong constraints, the agent would then first discard those packages including mountain climbing (that is, D^+c would be ignored), and among the remaining ones it would then look for packages satisfying at least some positive preferences.

If María selects DC_2, the formulae D^+c and D^-c are not allowed to have degrees summing up more than 1, and hence the above theory $\mathcal{T}''_{\mathcal{D}}$ becomes inconsistent. Actually, $\mathcal{T}''_{\mathcal{D}}$ becomes also inconsistent under DC_3, DC_3 is stronger than DC_2 (it does not even allow to have non-zero degrees for D^+c and D^-c). In these cases, the agent applies a revision mechanism, for instance to cancel $(D^+c, 0.85)$ from the theory.

Example 3 *(Example 2 continued)*

Suppose María's ankle is okay but she wants to travel with a very young nephew. In this situation she generates the desire of doing save activities (s) represented by the formula $(D^+s, 0.8)$. Moreover, considering the desire relation $D^+s \rightarrow_L D^+(\neg r)$, expressing that if a tourist prefers safe activities then he prefers not doing rafting is also in the agent theory, the agent infers $(D^+(\neg r), 0.8)$. Then, the extended theory is as follows:

$$\mathcal{T}'''_{\mathcal{D}} = \{(D^+m, 0.8), (D^+r, 0.6), (D^+(m \wedge r), 0.95), (D^+c, 0.85), (D^-f, 0.7), (D^+(\neg r), 0.8)\}$$

In the logic schema DC_1 it is not possible to have $(D^+r, 0.6)$ and $(D^+(\neg r), 0.8)$ in a consistent theory. Then, to restore the consistency one of these formulae must be forced to have desire degree equal 0. Instead, in schemas DC_2 and DC_3 there are no additional restrictions about this kind of formulae, and the agent may have both formulae maintaining consistency.

After analyzing in the previous subsections different schemas to model desires in an agent architecture, in the following section we show how these positive and negative desires may be used by the agent to generate intentions in the Intention Context (IC).

6.3 Intention Context

6.3.1 IC Language

We define in this context a suitable language to represent the agent's intentions. The syntax is defined in a similar way as we did with BC and DC, starting with a basic language $\mathcal{L}$ and incorporating a family of modal operators. In this case, in order to have a greater expressive power, we use the so-called Rational Łukasiewicz logic, RLL (see section 3.2.1), to represent and axiomatize the semantics of degrees of the intentions. RLL is an expansion of Łukasiewicz logic with a countable set of unary connectives $\{\delta_n\}_{n\in\mathbb{N}}$, whose intended semantics is that the truth-value of $\delta_n\varphi$ is just the truth-value of φ divided by n. So in RLL one can express divisions by natural numbers and these arithmetic operations are important in order to define the intention degree as weighted averages between diverse factors.

To define the IC Language we start from a basic propositional language $\mathcal{L}$. We assume the agent has a finite set of actions or plans Π^0 at her disposal to achieve

the desires. Then, for each $\alpha \in \Pi^0$ we introduce a modal operator I_α. such that the truth-degree of a formula $I_\alpha\varphi$ will represent the strength the agent intends φ by means of the execution of the particular action α.[5] We also introduce another modal operator I with the idea that $I\varphi$ will represent that the agent intends φ by means of the best plan in Π^0.

As in the other contexts, if the degree of $I_\alpha\varphi$ is δ, it may be considered that the truth degree of the expression "the desire φ is intended by means of plan α" is δ. If the degree of $I\varphi$ is γ, it may be considered that the truth degree of the expression "the desire φ is intended" is γ.

Therefore, many-valued $\mathcal{L}_{IC}$ formulae will be Rational Łukasiewicz logic formulae built from the set of propositional variables $Var_{cost} = \{C_\alpha\}_{\alpha\in\Pi^0}$ and elementary modal formulae $I_\alpha\varphi$ and $I\varphi$, where $\varphi \in Sat(\mathcal{L})$, and truth constants $\overline{r}$ for each rational $r \in [0, 1]$:

- If $\varphi \in \mathcal{L}$ then $\varphi \in \mathcal{L}_{IC}$
- If $\alpha \in \Pi^0$ then $C_\alpha \in \mathcal{L}_{IC}$
- If $\varphi \in Sat(\mathcal{L})$ and $\alpha \in \Pi$ then $I_\alpha\varphi, I\varphi \in \mathcal{L}_{IC}$
- If $r \in Q \cap [0, 1]$ then $\overline{r} \in \mathcal{L}_{IC}$
- If $\Phi \in \mathcal{L}_{IC}$ then $\delta_n\Phi \in \mathcal{L}_{IC}$
- If $\Phi, \Psi \in \mathcal{L}_{IC}$ then $\Phi \to_L \Psi \in \mathcal{L}_{IC}$ and $\neg_L\Phi \in \mathcal{L}_{IC}$

As usual, other Łukasiewicz logic connectives, like $\wedge_L, \vee_L, \equiv_L, \oplus_L, \otimes_L$ are definable from $\neg_L$ and $\to_L$.

We will call a (modal) formula *closed* when every propositional variable is in the scope of a I or a I_α operator.

The agent's intentions will be expressed by a theory $\mathcal{T}_\mathcal{I}$ (a set of closed formulae). Then, if the agent's IC theory $\mathcal{T}_\mathcal{I}$ contains the formula $I_\alpha\varphi \to_L I_\beta\varphi$ then the agent will try φ by executing the plan β before than executing plan α. On the other hand, if $\mathcal{T}_\mathcal{I}$ has the formula $I\psi \to_L I\varphi$ then the agent will try φ before ψ and it may not try ϕ if $(I\phi, \delta)$ is a formula in $\mathcal{T}_\mathcal{I}$ and $\delta < \tau$, where τ is an *intention threshold*.[6] This situation may mean for instance, that the benefit of getting ϕ is low or the cost is high.

[5] In the IC context we are not concerned about the question of whether a given desire can be reached by the execution of a particular action, this is left for the Planner Context, see Section 7.1.

[6] Set to discard the intentions with intention degree less than this value.

6.3.2 Semantics and axiomatization for IC

The semantics defined in this context shows that the value of the intentions depends on the formula intended to bring about and on the benefit the agent gets with it. It also depends on the agent's knowledge on possible plans that may change the world into one where the desire is true, and their associated cost. This last factor will make the semantics and axiomatization for IC somewhat different from the presented for positive desires in DC.

Models for IC are Kripke structures $M = \langle W, e, \{\pi_\alpha\}_{\alpha\in\Pi^0}\rangle$ where W is a set of worlds and $\pi_\alpha : W \times W \to [0,1]$ is the utility distribution corresponding to action α: $\pi_\alpha(w,w')$ is the utility of applying α to transform world w into world w'.[7] Further, $e : W \times (Var \cup Var_{cost}) \to [0,1]$ evaluates in each world propositional variables in such a way that variables from Var are evaluated into $\{0,1\}$ while propositional variables from Var_{cost} into $[0,1]$ (so variables from Var are Boolean while variables from Var_{cost} and many-valued). Then e is extended to Boolean formulae as usual and to atomic modal formulae by

- $e(w, I_\alpha\varphi) = \inf\{\pi_\alpha(w,w') \mid w' \in W, e(w',\varphi) = 1\}$
- $e(w, I\varphi) = \max\{e(w, I_\alpha\varphi) \mid \alpha \in \Pi^0\}$

and to compound modal formulae using the truth functions of Rational Łukasiewicz logic. Recall the interpretation of the δ_n connectives:
$e(w, \delta_n\Phi) = e(w,\Phi)/n$.

As usual, we will write $M \models \Phi$ when $e(\Phi, w) = 1$ for all $w \in W$ and will denote by $\mathcal{M}_{\mathcal{IC}}$ the class of all Kripke structures $M = \langle W, e, \pi_D, \{\pi_\alpha\}_{\alpha\in\Pi^0}\rangle$. Then, for each subclass of models $\mathcal{M} \subseteq \mathcal{M}_{\mathcal{IC}}$, given a theory T and a formula Φ, we will write $T \models_{\mathcal{M}} \Phi$ if $M \models \Phi$ for each model $M \in \mathcal{M}$ such that $M \models \Psi$ for all $\Psi \in T$.

The axiomatics for the IC logic is the following:

1. Axioms of classical logic for the non-modal formulae.
2. Axioms of Rational Łukasiewicz logic for the modal formulae, i.e. axioms of Łukasiewicz logic plus:
   $\delta_n\Phi \oplus \overset{n}{\ldots} \oplus \delta_n\Phi \equiv_L \Phi$
   $\delta_n\Phi \otimes (\delta_n\Phi \oplus \overset{n-1}{\ldots} \oplus \delta_n\Phi) \to_L \overline{0}$
3. (DC0) axiom for I_α modalities:
   $I_\alpha(\varphi \vee \psi) \equiv_L I_\alpha\varphi \wedge_L I_\alpha\psi$
4. Definitional Axiom for I:
   $I\varphi \equiv_L \bigvee_{\alpha\in\Pi^0} I_\alpha\varphi$

[7] Indeed, it can be seen as a kind of refinement of the R_α relations of the action dynamic logic semantics considered in the BC context.

5. Inference Rules:

   modus ponens for $\rightarrow$ and for $\rightarrow_L$

   introduction of I_α for implications: from $\varphi \rightarrow \psi$ derive $I_\alpha\psi \rightarrow_L I_\alpha\varphi$ for each $\alpha \in \Pi$.

The notion of proof for IC, denoted $\vdash_{IC}$, is defined as usual from the above axioms and inference rules. The presented axiomatics is obviously sound and one can prove completeness in an analogous way as for DC logic and we omit the proof.

Theorem 12 *Let T be a modal theory and Φ a modal formula. Then $T \vdash_{IC} \Phi$ iff $T \models_{\mathcal{M}_{IC}} \Phi$.*

It is worth noticing that so defined the semantics of the I_α operators is very general and probably it is not evident how to capture the idea that the truth-degree of a formula $I_\alpha\varphi$ should take into account not only how much φ is desired but also how costly is α. We define in the following, concrete $I\alpha$ operators as to show how the logical framework for the IC works.

Definition of concrete I_α operators

To take into account how much φ is desired and how costly is α, one can think of many possible ways. The possibly simplest one is to consider the value of $I_\alpha\varphi$ as the arithmetic mean between the value of $D^+\varphi$ and 1 minus the cost value of C_α. Suppose we have syntactically and semantically extended the language $\mathcal{L}_{DL}$ to receive $D^+\varphi$ formulae, coming from the DC by means of an appropriate bridge rule. Indeed, consider the following expression:

$$I_\alpha\varphi \equiv_L \delta_2 D\varphi \oplus \delta_2 \neg_L C_\alpha \qquad (I_\alpha\text{-Def})$$

Then, one can easily show that (I_α-Def) is consistent in IC. In fact, this formula is valid in all IC-models $M = (W, e, \pi^+, \{\pi_\alpha\}_{\alpha\in\Pi^0})$ such that π^+ is the same preference distribution defined in DC and $\pi_\alpha(w, w') = (\pi^+(w') + 1 - e(w, C_\alpha))/2$. Indeed, it holds that, for any $w \in W$,

$$\begin{aligned}
e(w, I_\alpha\varphi) &= \inf\{\pi_\alpha(w, w') \mid w' \in W, e(w', \varphi) = 1\} \\
&= \inf\{(\pi^+(w') + 1 - e(w, C_\alpha))/2 \mid w' \in W, e(w', \varphi) = 1\} \\
&= (\inf\{\pi^+(w') \mid w' \in W, e(w', \varphi) = 1\} + 1 - e(w, C_\alpha))/2 \\
&= (e(w, D\varphi) + e(w, \neg_L C_\alpha))/2 = \\
&= e(w, \delta_2 D\varphi \oplus \delta_2 \neg_L C_\alpha).
\end{aligned}$$

In other words, the above formula captures a notion of intention strength of reaching a desire φ through a plan α which is defined as the arithmetic mean of the desire degree of φ and of 1 minus the cost degree of action α. This notion of intention is at work in a bridge rule that will be described later.

Therefore, this axiom (or similar ones leading to different definitions for the I_α operators) can be included in a specialized theory over IC to specify particular behaviors of the Intention Context. Also, one can also specify some particular semantics for the plan cost variables C_α. For instance, one can consider the following natural axioms

(C1) $C_\gamma \equiv_L C_\alpha \vee C_\beta$, if $\gamma, \alpha, \beta \in \Pi^0$ and $\gamma = \alpha \cup \beta$

(C2) $C_\gamma \equiv_L C_\alpha \oplus C_\beta$, if $\gamma, \alpha, \beta \in \Pi^0$ and $\gamma = \alpha; \beta$

governing the costs of a nondeterministic union and of a concatenation of actions respectively. Axiom (C1) represents a kind of conservative attitude since it assigns to the nondeterministic plan $\alpha \cup \beta$ the maximum of the costs of α and β. Axiom (C2) establishes the (bounded) sum of costs of α and β as the cost of the plan $\alpha; \beta$. If we denote by IC_2 the extension of IC logic with these two axioms, then one can easily prove some consequences for the behavior of the I_α's operators in the theory defined by the (I_α-Def) formulae:

Lemma 13
(i) If $\alpha, \beta, \alpha \cup \beta \in \Pi^0$, *then* $\{(I_\gamma\text{−Def})\}_{\gamma \in \Pi^0} \vdash_{IC_2} I_{\alpha\cup\beta}\varphi \equiv_L I_\alpha\varphi \wedge I_\beta\varphi$
(ii) If $\alpha, \beta, \alpha; \beta \in \Pi^0$, *then* $\{(I_\gamma\text{−Def})\}_{\gamma \in \Pi^0} \vdash_{IC_2} I_{\alpha;\beta}\varphi \rightarrow_L I_\alpha\varphi \wedge I_\beta\varphi$

Proof: (i) comes from the fact that in RLL one can prove the following equivalences: $\neg_L(\Phi \vee \Psi) \equiv_L \neg_L\Phi \wedge \neg_L\Psi$, $\delta_n(\Phi \wedge \psi) \equiv_L \delta_n\Phi \wedge \delta_n\Psi$, and $\Gamma \oplus (\Phi \wedge \Psi) \equiv_L (\Gamma \wedge \Phi) \oplus (\Gamma \wedge \Psi)$. On the other hand (ii) is a consequence of the following implications provable in RLL: $(\Phi \rightarrow_L \Psi) \rightarrow_L (\delta_n\Phi \rightarrow_L \delta_n\Psi)$ and $(\Phi \rightarrow_L \Psi) \rightarrow_L (\Gamma \oplus \neg_L\Psi \rightarrow_L \Gamma \oplus \neg_L\Phi)$. □

6.4 Conclusions

We have presented a logical framework for the Desire and Intention contexts, both related with the representation of the agent's preferences. The DC represents the ideal preferences of the agent and the IC represents the goals the agent decides to follow after weighing the cost/benefit relation. In both cases we showed that the logical frameworks proposed are flexible enough to support different kinds of behaviors of these mental attitudes. Particularly, we notice that this kind of fuzzy-modal representation for desires and intentions allows to express in an explicit way qualitative expressions as for example $D^+\varphi \rightarrow D^+\psi$ and $I_\alpha\varphi \rightarrow I_\beta\varphi$ meaning respectively that we desire ψ more that φ, and we intend φ through the plan β with higher strength than through the plan α.

On the one hand, different proposals for modelling desires in DC have been presented from a basic DC schema. The most suitable alternative may be chosen for defining particular agents. The purpose of presenting different consistency schemas was to show that the framework presented for the agent Desire context is capable

to represent different kinds of bipolar and graded preferences, as for example, the model proposed by Benferhat et al [12].

On the other hand, in the IC we have defined a general logical context and then, we showed how a concrete definition of the Intention through a plan, involving different factors, as for instance the desire degree and cost of the plan, may be defined in a consistent way. Other similar definitions of the intention degree representing the way the agent takes the decisions are possible.

In general, we have followed a "blind" conception of desires where the preference relations (positive or negative) over formulae are translated into preferences over worlds considering what happens in the different worlds only with respect to that formulae, independently of what happens with the rest (i.e. the values that take the other formulae in the worlds), for instance if we have $D^+A \rightarrow D^+B$ (that is 1-true whenever $D^+A \leq D^+B$) semantically means that the preference measure over the worlds where A is satisfied is greater than the measure over the worlds where B is true, but nothing is said about the rest of the formulae in the worlds w where A or B are satisfied. Another approach following the *Ceteris Paribus principle* may be interesting to analyze, where the relations over formulae like $D^+A \rightarrow D^+B$ may be transfered to worlds where the only difference between them is about A and B and the rest remaining the same. This approach is left as future work.

With respect to the proposed axiomatics to model desires and intentions in the BDI logic (K and D axioms), the K axiom is covered by the current axiomatics presented respectively for DC and IC. Considering the D axiom $GOAL\phi \rightarrow \neg GOAL(\neg\phi)$ for desires (goals), in the DC logic we did not include any kind of restriction over the desires on a formula and its negation, but we formalized this constraint in a many-valued framework, in the schema DC_1 by adding the axiom $(D^+\varphi \wedge_L D^+(\neg\varphi) \rightarrow_L 0)$

Preliminary work related to desire and intention contexts in the g-BDI agent model can be seen in [29, 30, 31]. Recently, in [37] we have presented the logical framework to represent and reason about graded preferences and intentions.

An important question related to preference modelling is how to use preferences in order to get the best solutions. In the case of our agent architecture, this question turns to: how can the agent use positive and negative desires to get the best intentions that in turn lead him to plans. There are different ways of using (or combining) positive and negative preferences in order to find the best solutions, for instance Benferhat et al. present different approaches in [12]. In the g-BDI model of agent the positive and negative preferences are pro-active attitudes that guides the search of which is the best intention the agent may follow and suitable bridge rules are defined to combined in a flexible way this preference information with other elements (e.g., the plan cost, the belief degree of achieving the goal by plan execution) to decide the best intention the agent may follow through a selected plan. In the next Chapter 7 we introduce this kind of bridge rules.

The whole is more than the sum of the parts.

Aristotle

Chapter 7

Functional contexts and Bridge rules

In this Chapter we present the remain necessary components of our multi-context agent model: the Planner context (PC), the Communication context (CC) and the Bridge rules (BR). Finally, a simple example to show how our agent model works is presented.

7.1 Planner and Communication Contexts

The nature of these contexts is functional. The Planner Context (PC) has to build plans which allow the agent to move from its current world to another, where a given formula is satisfied. This change will indeed have an associated cost according to the actions involved. Within this context, we propose to use a first order language restricted to Horn clauses (PL), where a theory of planning includes the following special predicates:

- *action(α, P, A, c_α)* where $\alpha \in \Pi^0$ is an elementary action, $P \subset PL$ is the set of preconditions; $A \subset PL$ are the postconditions and $c_\alpha \in [0, 1]$ is the normalized cost of the action.

- *plan(φ, α, P, A, c_α)* where $\alpha \in \Pi$ is a composite action representing the plan to achieve φ, P are the pre-conditions of α, A are the post-conditions, $\varphi \in A$ and c_α is the normalized cost of α.

- *bestplan(φ, α, P, A, c_α)* similar to the previous one, but only almost one instance with the best plan is allowed.

The Planner context is in charge of looking for a set of plans called *feasible plans*: *f-plan(φ, α, P, A, c_α)*. In this PC approach, these plans are generated to fulfill positive desires (φ if $(D^+\varphi, d)$ and $d > 0$), the current state of the world w must satisfy the preconditions (for all $\phi \in P$: $w \models \phi$), the plan must make true

the positive desire the plan is built for ($\varphi \in A$), but avoiding negative desires (i.e. cannot have any negative desire as post-condition $A \not\models \psi$ if $(D^{-}\psi, n)$ and $n > 0$.

Furthermore, a filter may be used to select those plans that achieve the desire φ with a belief degree greater than some threshold (τ_b): $(B([\alpha]\varphi), \tau_b)$. In the representation of a feasible plan the normalized cost of the plan, $c_\alpha \in [0, 1]$, is also included.

The communication unit (CC) makes it possible to encapsulate the agent's internal structure by having a unique and well-defined interface with the environment. This unit also has a first order language restricted to Horn clauses. The theory inside this context will take care of the sending and receiving of messages to and from other agents in the Multi-agent society where our graded BDI agents live. Also, through the communication unit the agent perceives changes in the environment. Depending on the kind and the source of the information, an uncertainty degree may be associate to the incoming data. This issue is in close relation to the trust in the information source and some view of this problem is analyzed in next Chapter 8. All the information that the CC receives will be introduce to the Belief context BC by a suitable bridge rule (see (7.5) in next Section 7.2).

Also, the CC is in charge of communicating the action the agent choses to execute and this is done by another bridge rule (see (7.4) in next Section 7.2).

Both functional contexts use resolution as a deduction method.

7.2 Bridge Rules

The deduction mechanism of multi-context systems is based on two kinds of inference rules, internal rules, inside each unit; and bridge rules, outside. Internal rules allow to draw consequences within a theory, while bridge rules allow to embed results from a theory into another (see Section 2.3 for details). Then, bridge rules can be understood as rules of inference with premises and conclusions in different contexts. A multi-context systems needs some kind of control strategy as to prevent that a bridge rule executes more than once under the same premise conditions.

For our g-BDI agent model, we define a collection of basic bridge rules to establish the necessary interrelations between context theories. Some of these rules are illustrated in figure 11.6. In this Section we comment some of the most relevant Bridge rules schemas:

1. There are bridge rules from DC to PC that, from the positive and negative desires (pro-active attitudes), generate predicate instances in the PC unit that are used by the planner program to build the feasible plans:

$$\frac{DC : (D^{+}\varphi, d)}{PC : \lceil (D^{+}\varphi, d) \rceil} \quad \text{and} \quad \frac{DC : (D^{-}\psi, n)}{PC : \lceil (D^{-}\psi, n) \rceil} \tag{7.1}$$

2. The agent knowledge about the world state and about actions that change

the world, is introduced from the belief context into the Planner as first order formulae:

$$\frac{BC : B\varphi}{PC : \lceil B\varphi \rceil} \tag{7.2}$$

3. Regarding intentions, there is a bridge rule that infers the degree of $I_\alpha\varphi$ for each feasible plan α that allows to achieve φ. The intention degree is thought as a trade-off among the benefit of reaching a desire, the cost of the plan and the belief degree in the full achievement of φ after performing α. The following bridge rule computes this value from the degree of $D^+\varphi$ (d), the degree of belief $B[\alpha]\varphi$ (r), the cost of the plan α (c):

$$\frac{DC : (D^+\varphi, d), BC : (B[\alpha]\varphi, r), PC : fplan(\varphi, \alpha, P, A, c)}{IC : (I_\alpha\varphi, f(d, r, c))} \tag{7.3}$$

   Different functions f allow to model different agent behaviors. For instance, if we consider an *equilibrated agent*, where all the factors involved are equally taken into account, the function might be defined as the average among these factors. In other cases, a weighted average may be used where the different weights w_i are set according to the agent expected behavior:

$$f(d, r, c) = (w_d d + w_r r + w_c (1 - c)) / (w_d + w_r + w_c)$$

   For example, for a *greedy agent*, w_c may be set greater than the other weights: w_d and w_r.

4. The information supplied by the above bridge rule to the IC unit allows this unit to derive, for each desire φ, a formula $(I\varphi, i)$ where i is the maximum degree of all the $(I_\alpha\varphi, i_\alpha)$ formulae, where α is a feasible plan for φ. The plan α_b that allows to get the maximum intention degree i_{max} considering all the agent desires, will be set by the PC unit as the *best plan* (see the definitional axiom for I in Section 6.3). Finally, we also need rules to establish the agent interaction with the environment, meaning that if the agent intends φ at degree i_{max}, the maximum degree of all the intentions, then the agent will choose to execute the plan α_b —*bestplan*— that will allow him to reach the most intended goal φ:

$$\frac{IC : (I_{\alpha_b}\varphi, i_{max}), PC : bestplan(\varphi, \alpha_b, P, A, c)}{CC : C(does(\alpha_b))} \tag{7.4}$$

5. Through the communication unit the agent perceives all the changes in the environment that are introduced by the following bridge rule in the belief context:

$$\frac{CC : \beta}{BC : B\beta} \tag{7.5}$$

Bridge rules to represent realism

In BDI agents, bridge rules have been also used to determine the relationship between the mental attitudes and the actual behavior of the agent [115]. Well-established sets of relations for BDI agents have been identified called realism models [41, 125].

- *Strong Realism:* the set of intentions is a subset of the set of desires, which in turn is a subset of the beliefs. That is, if an agent does not believe that something may became true, it will neither desire it nor intend it:

$$\frac{BC : \neg B\psi}{DC : \neg D\psi} \text{ and } \frac{DC : \neg D\psi}{IC : \neg I\psi} \tag{7.6}$$

- *Realism:* The set of the agent beliefs is a subset of the agent desires which in turn is a subset of the set of intentions. That is, if an agent believes something, she both desires and intends it.

$$\frac{BC : B\psi}{DC : D\psi} \text{ and } \frac{DC : D\psi}{IC : I\psi} \tag{7.7}$$

- *Weak Realism:* This is a middle point between strong realism and realism. An agent do not desire a property if its negation is believed, does not intend something if its negation is desired, and does not intend something if its negation is believed.

$$\frac{BC : B\psi}{DC : \neg D\neg\psi} \text{ , } \frac{DC : D\psi}{IC : \neg I\neg\psi} \text{ and } \frac{BC : B\psi}{IC : \neg I\neg\psi} \tag{7.8}$$

Notice that in these rules we present a simplified version of these relations were the time dimension is not taken into account as it is the case in [115]. Some work to overcome this limitation was proposed in [136] introducing the notion of time in Bridge rules of a multi-context agent, particularly to represent the time consumed in passing formulae and action execution.

Bridge rules to generate desires in a dynamic way

In the desire context DC different schemas to represent and reason about desires were presented but how desires are derived was not discussed. In a dynamic environment the agent desires will change, depending on her beliefs and also on the set of current desires.

Notably, Rahwan and Amgoud in their argumentation-based approach to practical reasoning [122] provide an argumentation-based framework for generating consistent desires, among other tasks (see Section 2.5 for details). The basic elements of this argumentation framework are the desire-generation rules, as follows:

- *Desire-Generation Rule* or a desire rule, is an expression of the form:

  $\varphi_1 \wedge ... \wedge \varphi_n \wedge \psi_1 \wedge ...\psi_m \Rightarrow \psi$ where $\varphi_i \in \mathcal{K}$ and $\psi_j, \psi \in \mathcal{D}$. The set $\mathcal{K}$ represents the agent knowledge and the set $\mathcal{D}$ gathers all possible agent desires.

  The meaning of the rule is "if the agent believes $\varphi_1, ..., \varphi_n$ and desires $\psi_1, ..., \psi_m$, then the agent will desire ψ as well".

These rules are also similar to the filtering rules proposed in [132] to represent reasons for and against adopting desires.

Thus, we can introduce in our g-BDI model a multi-context and many-valued version of these rules. As the desire and belief formulae in the premise are coming from different contexts, we define the following bridge rules for desire generation:

$$\frac{BC : (B\varphi_1 \wedge ... \wedge B\varphi_n, b), DC : (D^+\psi_1 \wedge ... \wedge D^+\psi_m, c)}{DC : (D^+\psi, d)} \tag{7.9}$$

Namely, if the agent has the beliefs $B\varphi_1, ..., B\varphi_n$ in degree greater or equal then a threshold b and positively desires $D^+\psi_1, ..., D^+\psi_m$ in degree at least c, she also desires ψ in degree at least d.

With the description of this set of bridge rules (BR) we have finished the presentation of all components of the g-BDI agent model. Below we give some insights of how the agent model works and present a simple example as to show how these components interact to decide the agent action. Then, in next Chapter 12 we develop a case study extracted from a real world domain.

7.3 How the g-BDI model works

Up to this point we have proposed an expressive logical framework to represent and reason about an agent's beliefs, desires and intentions.

In order to make all the described logical ingredients operational in a deliberative agent architecture, they should be complemented with the proposed functional contexts (Communication Context and Planner contexts) and some bridge rules, which allow to pass formulae among theories.

We describe next the working of the agent architecture its main components, which are shown in Figure 7.1 where circles represent the different contexts and their theories, boxes represent tasks carried out by special bridge rule inferences, and where arrows illustrate the information flow and context interaction by bridge rules.

1. *CC:* receives the environment input.
2. *Information passes from CC to BC* (see BR 7.5).
3. *BC:* represents the uncertain information the agent has about its environment.

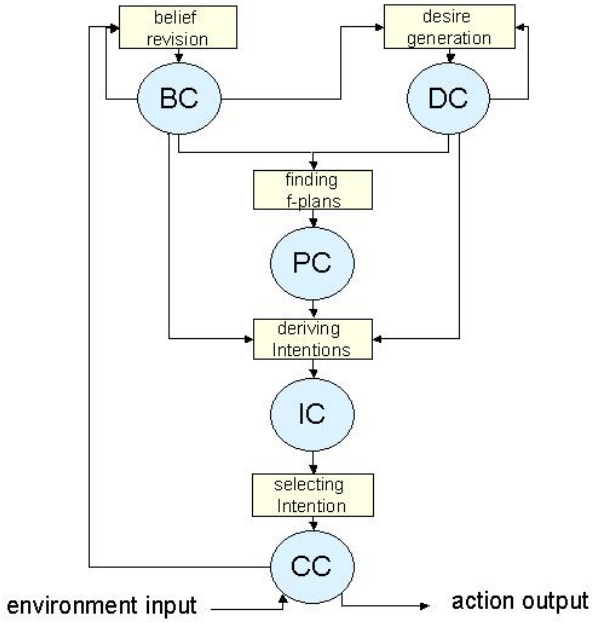

Figure 7.1: Agent architecture

4. *DC:* represents the graded positive and negative agent desires.

5. *A belief revision process*, which takes new inputs (from CC) and the agent's current beliefs (from BC) and determines a new set of graded beliefs.[1]

6. *A desire generation and revision process*, which determines and/or revises the agent's graded positive and negative desires on the basis of her current beliefs and her previous desires (see BR 7.9).

7. *Desires are passed from DC to PC* (see BR 7.1).

8. *Beliefs about plans and domain knowledge are passed form BC to PC* (see BR 7.1).

9. *PC: looks for feasible plans*,feasible plans are plans which fulfill (to some degree) positive desires, satisfy some preconditions and avoid undesired postconditions. Filtering plans, to identify which ones are feasible, can be achieved in

[1]Even thought in this approach we haven't included some revision processes for the different contexts, we consider they are necessary for an agent that lives in a dynamic environment

a logic style by the following rule:[2]

$$\frac{\begin{array}{c}(D^{+}\varphi, r), (D^{-}\psi, s), plan(\alpha, P, A, c),\\ (B([\alpha]\varphi), b_0), (B(P), 1), (B(A \rightarrow \neg\psi), 1)\end{array}}{fplan(\varphi, \alpha, P, A, c)} \quad (1)$$

which generates a predicate *fplan(φ, α, P, A, c_α)*, standing for α is plan that achieves φ with precondition P, postcondition A and cost c, whenever: (1) it is believed (above some threshold level b_0) that plan α leads to satisfy a positively desired goal φ (encoded as $(D^{+}\varphi, r)$ and $(B([\alpha]\varphi), b_0)$), (2) α's precondition P is satisfied $((B(P), 1))$, and (3) α's postconditions A avoid negative desires ψ (encoded as $(B(A \rightarrow \neg\psi), 1)$ and $(D^{-}\psi, s)$).

10. *PC:* has a set of feasible plans (fplans) according to the agent's beliefs and desires.

11. *A process for deriving intentions*, which for each feasible plan α that allows to achieve a desire φ, an intention formula $I_\alpha\varphi$ is derived with its corresponding degree. According to the notion adopted the intention degree is taken as a trade-off between the benefit of reaching the desire and the cost of the plan α. This can be made operational by means of an inference rule like the one showed in previous Section (see BR 7.3) and using as the function f a weighted average as it was proposed.

12. *IC:* represent the set of current graded Intentions, representing those desires that the agent is committed to try to bring about by the execution of feasible plans (see BR 7.3).

13. *Intention - Action selection process*, which determines which action to perform on the basis of each selected intention. From the set of current intentions and feasible plans, the agent selects for a given desire φ the plan α which leads to a maximum intention degree for $I_\alpha\varphi$, represented by the degree of the formula $I\varphi$.

14. *CC: communicates the action α the agent undertakes.*

We illustrate how the g-BDI agent model works by the following example.

7.4 Example

(Continuation of Example 1, see Chapter 6) We have a Tourism Recommender Agent in charge of looking for different tourism plans in order to satisfy a set of

[2]Notice that this is not a BR, is an internal rule in the PC that uses formulae that have been injected by BRs (agent beliefs and desires).

tourist preferences. She will intend to reach the goal (i.e. satisfying the user) by the recommendation of the tourism plan which can get her the highest intention degree. The recommender agent is modelled using the g-BDI architecture and takes all desires expressed by María, our stressed tourist (also presented in Example 1), and following the steps explained in this section, finds the best recommendation for María:

(1-2-3) **Update Belief Context:** the agent updates her current beliefs about the tourism plans offered, the tourism domain (structured using destinations ontologies) and the beliefs about how these packages can satisfy the user's preferences. All these beliefs constitute the agent current belief theory T_B.

(4-6) **Update Desire Context:** from María preferences the agent generates her desires exactly at they were generated in Example 1 and are represented as the desire theory: $T'_D = \{(D^+m, 0.8), (D^+r, 0.6), (D^+(m \wedge r), 0.95), (D^-f, 0.7)\}$.

Once these theories are defined, the Agent is ready to reason in order to determine which Intention to adopt and which plan is associated with that intention.

(7) The desires are passed from DC to PC.

(8) The beliefs about tourist packages and tourism are passed from BC to PC.

(9) **PC - looking for feasible packages:** from this set of positive and negative desires (T'_D) and knowledge about the tourist packages the agent can offer and the benefits they bring (T_B), and using a Planner, the agent looks for feasible plans, that are believed to achieve positive desires (i.e in this case m, r or $m \wedge r$) by their execution but avoiding the negative desire (i.e. f) as postcondition (see rule (1)).
The agent has in its knowledge base among other formulae, the following ones representing the description of the offered tourism plans. Each package is represented by predicates defined as: $Package(Id, null, Trip, Cost)$

where Id is the identification of the plan, we consider null preconditions, $Trip$ is a detailed travel-stay sequence (postconditions) and $Cost$ is the package cost.

The Planner looks for the plans that do not satisfy f and that satisfies m or r, by a cross-searching between the package and destination ontologies.

(10) **Current feasible packages:** the agent finds that the plans *Mendoza* (Me) and *SanRafael* (Sr) are feasible plans for the combined desire $m \wedge r$, while *Cumbrecita* (Cu) is feasible only for m. The Planner also computes the normalized cost ($c \in [0, 1]$) of these plans being respectively: $c_{Me} = 0.60$ and $c_{Sr} = 0.70$ and $c_{Cu} = 0.55$.

The agent also has the following beliefs related to the achievement of the different desires (m, r or $m \wedge r$) by the feasible plans (Me, Sr and Cu). These beliefs are included in the BC theory.

$$T'_B = \{(B[Me]m, 0.7),\ (B[Me]r, 0.6),\ (B[Me](m \wedge r), 0.6),\ (B[Sr]m, 0.5),\ (B[Sr]r, 0.6),\ (B[Sr](m \wedge r), 0.5),\ (B[Cu]m, 0.4),\ (B[Ba]m, 0.8),\ (B[Ba](f), 1)\}$$

(11) **Deriving the Intention formulae $I_\alpha\varphi$, for each feasible plan α towards a desire φ.** The intention degrees for satisfying each desire m, r and $m \wedge r$ by the different feasible plans are computed by a rule that trades off the cost and benefit of satisfying a desire by following a plan. The Agent uses rule 7.3 (Section 7.2) and considering the function f as the average of the desire degree (d), the belief degree r of achieving the desire by a selected plan and the complement of the cost (c):

$$f(d, r, c) = (d + r + (1 - c))/3$$

computes the intentions degrees towards $m \wedge r$, m and r by executing the feasible plans *Mendoza* (Me) and *SanRafael* (Sr).

(12) **Current Intentions:** as a result of the previous process, the set of intentions contains the following formulas:

$(I_{Me}(m \wedge r), 0.675)$, $(I_{Sr}(m \wedge r), 0.625)$,
$(I_{Me}(m), 0.60)$, $(I_{Me}(r), 0.50)$,
$(I_{Sr}(m), 0.55)$, $(I_{Sr}(r), 0.45)$,
$(I_{Cu}(m), 0.625)$.

(13-14) **Selecting Intention-plan:** the agent decides the tourist recommendation. From this set of current intentions, the Agent decides to recommend the plan *Mendoza* (Me) since it brings the best cost/benefit relation (represented by the intention degree 0.675) to achieve $m \wedge r$, satisfying also the tourist negative desire.

7.5 Comparison with Rahwan and Amgoud's approach

In this Chapter we have completed the formalization of our g-BDI agent model by presenting the functional contexts and the bridge rules. Also, we have described the working of the proposed architecture. Then, we are able to compare this agent model with a relevant related approach.

Previously, in Chapter 2.5, we have analyzed different approaches to graded attitudes in intentional agents. Noticing that most proposals related to graded beliefs, weighted preferences and intention reconsideration, model partial aspects of the uncertainty related to mental notions involved in agent architectures. Notably, Rahwan and Amgoud have recently presented in [122], a complete argumentation-based framework for practical reasoning. They provide a rich argumentation-based framework for (i) generating consistent desires, and (ii) generating consistent plans for achieving these desires. We belief that our agent model is complementary to their approach in different aspects we next detail.

This argumentation-based approach allows the representation of uncertain beliefs and worth related to desires. In this work, however, the authors do not present strictly speaking a formal system (in the sense of a logical system which is sound and complete with respect to an intended semantics) to represent and reason with these graded attitudes according to a suitable uncertainty model. The relation between their proposal and some BDI axiomatics is left as future work. In this direction, we have developed well founded logics to model the different mental attitudes (i.e. beliefs, desires and intentions) in a multi-context framework.

Besides, Rahwan et al. in [122] do not use any estimation on plan failure and have no estimation on the uncertainty of achieving the desire after the plan execution. They work with certain plan rules. In this sense, our proposal includes a preliminary approximation to the notion of plan failure by computing the belief degree in having the goal after executing a determine plan (i.e. by using $B[\alpha]\varphi$ formulae).

In their model, they select intentions from a set of justified and feasible desires, constructing a complete pre-order relation, where the worth of the desires in the plan, and the cost of its resources are involved. To decide intentions they present an interest combination based on utility, as the difference between of total worth of desires and the total cost of the resources involved in their achievement. For the g-BDI model we have proposed a process to decide the agent current intention by looking for the desires that can be achieved by feasible plans (similar to their notion of feasible desires) and then, the best ranked intention is selected (see Chapter 6). This ranking is built by using a function that combines among others, the desire degree and cost of the plan. Instead of specifying a fixed combination, we proposed that different functions may be defined to model different agent behaviours.

We remark they have presented an important approach to the deliberation process in BDI agents towards desire generation. They include desire generation rules that from desires and belief, generate new desires. We have incorporated these rules to our agent model to generate desires in a dynamic domain, by defining suitable bridge rules (7.9 in Section 7.2). Furthermore, the argumentation frameworks they propose for belief, desires and plans, permit to treat in an efficient way with the agent inconsistent information. In this way their argumentation model maintains the different bases sound and it allows to look for a conflicting free set of acceptable plans. We consider that the argumentation based approach is a promising direction for future work related to the revision of the different attitudes in our g-BDI model.

Trust is the lubrication that makes it possible for organizations to work.

W. G. Venís

Chapter 8

The Socialization of the g-BDI agents

8.1 Introduction

The agents developed using the g-BDI model may interact with the environment and other agents, human or not, in the agent society where they are situated. Thus, it is necessary to take into account the social aspects of agency. The agent interactions have many facets and much work still must be done in different directions. In this chapter we consider some preliminary steps towards the g-BDI agent socialization so as to show how the agent model can be extended to include some of these aspects.

First, we show how the language defined for the mental contexts can be used to reason about other agent attitudes. Second, a *social context* is added to the g-BDI agent model to represent and handle different kinds of trust in other agents. Particularly we consider the trust in informant agents and the trust in delegating plans to other agents.

8.2 Reasoning about other agent attitudes

The languages defined in previous chapters for the different mental contexts (BC, DC and IC) in the g-BDI agent model include modal formulae over a base propositional language $\mathcal{L}$, i.e. formulae of the kind $M\varphi$, where $\varphi \in \mathcal{L}$ and $M \in \{B, D, I\}$. These languages do not permit nested modalities, i.e. formulae of the kind $BB\varphi$ or $BI\psi$ are not allowed. This is actually a language limitation for an agent that may need to reason about other agents' beliefs, desires or intentions, and would need to use formulae like $B_a B_b \varphi$ or $B_a I_b \psi$, expressing that agent a believes that agent b believes φ or that agent a believes agent b intends ψ. The problem with this more general formulae is, first of all, that they cannot be given a suitable meaning by the different kinds of Kripke structures we have defined for the different context logics. Namely, let us consider for instance the logic BC_{prob}. In a BC_{prob} Kripke structure, a formula

$B\varphi$ is evaluated by a probability measure μ defined on (classical) subsets of worlds. If we would like to evaluate a formula like $B(B\varphi)$, then notice that $B\varphi$ is a many-valued formula (it takes a value from $[0, 1]$ in each model) and hence the outermost B operator should be evaluated over a fuzzy subset of worlds.

To overcome this problem, a partial solution would be to force that modal operators B, D and I apply over *Boolean* modal formulae, in the sense that they either take degree 1 or degree 0 in each world. This can be achieved by first expanding the fuzzy logics used in the different contexts (e.g. Łukasiewicz or Gödel logic) with the projection Baaz Δ operator (see Section 3.2), with the following interpretation:

$$e(\Delta\varphi) = \begin{cases} 1, & \text{if } e(\varphi) = 1 \\ 0, & otherwise \end{cases}$$

From Δ we can define other projection operators, for example the operator $\nabla = \neg\Delta\neg$ [78], evaluated as follows:

$$e(\nabla\varphi) = \begin{cases} 1, & \text{if } e(\varphi) > 0 \\ 0, & otherwise \end{cases}$$

Notice that although φ may be a fuzzy formula, $\Delta\varphi$ and $\nabla\varphi$ are crisp (bi-valued). Therefore, if we stipulate that every modal operator can only be applied to either a propositional formula or to a modal expression beginning with Δ or ∇, then we could safely use and evaluate and agent's beliefs about other agents' attitudes provided that BC_{prob} Kripke structures $M = (W, \mu, e)$ are extended to structures of the kind $M = (W, \mu_a, \mu_b, \ldots, e)$, with a probability measure for each agent. For instance, in such a case, we could represent and evaluate expressions like:

- $B_a(\nabla I_b\varphi)$ representing that "the agent a believes that the agent b intends φ in some positive degree".
- $\overline{0.6} \rightarrow B_a(\nabla B_b\varphi)$ expressing that " the agent a believes with a degree greater or equal than 0.6, that the agent b believes φ with a positive degree".
- $\overline{0.7} \rightarrow B_a(\Delta(\overline{0.5} \rightarrow D_b\varphi))$ meaning that "the agent a believes with a degree greater or equal than 0.7, that the agent b desires φ with a degree greater or equal than 0.5".

These restricted nested formulae work properly, in principle, to combine different modalities from a semantic point of view. However, this approach does not improve the language expressive power. For example, consider the formula $\overline{0.7} \rightarrow B_a(\Delta(\overline{0.5} \rightarrow B_b\varphi))$. Since the probability measures μ_a and μ_b do not depend on the particular world we are but on the whole model, then $\overline{0.5} \rightarrow B_b\varphi$ either holds or not in any world of W, and hence $B_a(\Delta(\overline{0.5} \rightarrow B_b\varphi))$ gets either value 1 or 0, and so, it is trivially true or false, and the complex belief formula could actually be equivalently represented by the formula $\Delta(\overline{0.5} \rightarrow B_b\varphi)$, without using any nesting. Therefore, to get a better expressive power, the semantics should also be extended

by considering probabilistic Kripke structures of the kind $M = (W, \mu_a, \mu_b, \ldots, e)$ where now $\mu_i : W \times F_i \rightarrow [0, 1]$, such that for each world $w \in W$, $\mu_i(w, \cdot)$ is a probability measure over some suitable Boolean subalgebra $F_i \subseteq 2^W$.

The full development of these kinds of more expressive power is left for future work.

8.3 Trust in an agent society

To equip an agent with tools in a social context, it is important to model and support the agent's trust. In an agent community different kinds of trust are needed and should be modelled, as it was pointed out in [1, 39]. They can be grouped in trust in the environment and trust in other agents interacting in it (e.g., mediating agents, potential partners). Trust is a mental state, a complex social attitude of an agent x toward another agent y about the belief in the behavior/action relevant for a goal. Castelfranchi and Falcone in [39] have developed a complete cognitive approach to social trust.

The use of previous direct interactions is probably the best way to calculate trust but, unfortunately, this information is not always available. Reputation systems take advantage, among other things, of social relations between agents to overcome this problem as is discussed in [137]. A valuable example of this kind of tools is the ReGret system [135] where the reputation of the participating agents is modelled taking into account diverse social relations in a trading context.

To deal with the social aspects of agency in our g-BDI agent model we introduce a new context, the Social Context (SC), with the purpose of modelling the agent trust or reputation in other agents. In the multi-context specification of the g-BDI agent model, besides the *mental* contexts to represent beliefs (BC), desires (DC), intentions (IC), and two *functional* contexts, for Planning (PC) and Communication (CC); a social context (SC) is thus included:

$$A_g = (\{BC, DC, IC, SC, PC, CC\}, \Delta_{br})$$

The interrelation of the SC with the other contexts in the architecture can be defined in a neatly way, by suitable bridge rules.

Among the different kinds of trust that may be used in an agent society, in this chapter we consider two cases. On the one hand, the incoming information must be analyzed and filtered depending on the trust that the agent has in its source. On the other hand, the social trust must also be considered when the agent must decide about the delegation to others of an action that is part of the plan toward the goal that it is trying to achieve.

8.3.1 A Social Context to Filter Information

In a first stage, we consider that the purpose of the social context (SC) in our agent model is to filter all the information coming from other agents. We have inspired our work in the Belief, Inform and Trust (BIT) logic presented by Liau [98] and then, extended by Dastani [43]. One of the central ideas formalized in BIT logic is the following: "if $agent_i$ is informed by $agent_j$ about φ, the $agent_i$'s beliefs about φ depends on the trust the $agent_i$ has in $agent_j$ with respect to φ". In the framework of this logic all the formulae are crisp. We extend this idea to a many valued approach, in a multi-context specification. Preliminary results in this direction can be seen in [31].

Language

Assuming we have a multiagent system scenario with a finite set of agents: $\{agent_i\}_{i \in A_g}$, the language for this social context $\mathcal{L}_{SC}$ is built over a basic propositional language $\mathcal{L}$, extended by a family of modal operators T_{ij} where $i, j \in A_g$. Then, the language $\mathcal{L}_{SC}$ contains non-modal $\mathcal{L}$-formulae and modal formulae $T_{ij}\varphi$, where $\varphi \in \mathcal{L}$. As in the definition of the DC language (see Section 6.2), we call a modal formulae *closed* when every propositional variable is in the scope of a modal operator T_{ij}.

The formula $T_{ij}\varphi$ represents the trust of $agent_i$ towards $agent_j$, with respect to φ. We consider that these formulae may be graded taking values in [0,1], to express different levels of trust. A belief-based degree of trust has been discussed in [39].

In the same way than in the other mental contexts, we use a many-valued treatment for the trust of an agent in others. Then, if the degree of $T_{ij}\varphi$ is τ, we shall consider that the truth degree of the sentence "$agent_i$ trusts in $agent_j$ about φ" is τ. We also choose the Łukasiewicz logic as the underlying many-valued logic.

In this context, the agent trust in other agents will be represented by a theory $\mathcal{T}$ (a set of closed $\mathcal{L}_{SC}$ formulae) containing quantitative expressions about the agent trust, as for example $(T_{ij}\varphi, \tau)$ and $(T_{ik}\psi, \gamma)$.

Semantics and axioms

The models for SC are defined in a similar way as we did in the other contexts using a Kripke structure. $M_S = \langle W, e, \{\tau_{ij}\}_{i,j \in A} \rangle$ where W and e are defined in the usual way, and for each $i, j \in A$, $\tau_{ij} : W \to [0, 1]$ is a trust distribution over worlds where $\tau_{ij}(w) \in [0, 1]$ is the degree on which $agent_i$ trusts $agent_j$ about the possibility of w being the actual state of the world.

The truth evaluation $e : V \times W \to \{0, 1\}$ is extended to the non-modal formulae in the usual way. For the modal formulae, we follow the intuition that the trust on $\varphi \wedge \psi$ may be taken as the minimum of the trust on φ and on ψ, hence we interpret the trust operator T_{ij} as a necessity measure on non-modal formulae. We extend the truth-evaluation e to modal formulae using Łukasiewicz logic truth-functions and by

defining:

$$e(T_{ij}\varphi, w') = \inf\{1 - \tau_{ij}(w) \mid w \in W, e(w, \varphi) = 0\}$$

that is, the necessity of φ with respect to the possibility distribution τ_{ij}.

Then, the corresponding axiomatics is set in a similar way than in the basic schema for DC, as follows:

1. Axioms of classical logic for the non-modal formulae,
2. Axioms of Łukasiewicz logic for the modal formulae,
3. axiom for T_{ij} modalities:
   (SC1) $T_{ij}(\varphi \wedge \psi) \equiv_L T_{ij}\varphi \wedge_L T_{ij}\psi$
   (SC2) $\neg_L T_{ij}\bot$
4. Inference Rules:
   (MP) modus ponens for $\rightarrow$ and for $\rightarrow_L$
   (EQ) from $\vdash_{CPC} \varphi \equiv \psi$ derive $\vdash T_{ij}\varphi \equiv_L T_{ij}\psi$
   necessitation rule for each T_{ij} where $i, j \in A_g$: from φ derive $T_{ij}\varphi$.

After the trust between the agent and the other agents in the environment respect to a subject is defined, the agent can use this notion of trust to asses the quality of the information received from other agents. Following [43] we want to use this trust to derive the agent beliefs about what is being informed.

In a multiagent system scenario, if $agent_i$ is informed by $agent_j$ that φ is true, this statement may be represented by a first order predicate:

$$\mathit{informed}(agent_i, agent_j, \varphi) \text{ (two-valued formula represented by } N_{ij}\varphi).$$

The main axiom for trust in the BIT logic [98] where the agent filter behavior is set, is defined as follows:

(C1) $(B_i N_{ij}\varphi \wedge T_{ij}\varphi) \rightarrow B_i\varphi,\ \mathit{where}\ i, j \in A_g$ (filter)

We present a multi-context version of this axiom. As belief, information and trust formulae are represented in different contexts, we use a special bridge rule to formalize it and we extend this rule to a many-valued framework.

Through the Communication unit —CC (outlined in Chapter 7), the agent perceives all the changes in the environment and, in particular, it receives the information from other agents (e.g. $N_{ij}\varphi$ formulae). Then, using the trust degree (τ) on $agent_j$ with respect to φ, this information is introduced in the belief context, using a suitable order preserving transformation h, by the following bridge rule:

$$\frac{CC_i : N_{ij}\varphi, SC_i : (T_{ij}\varphi, \tau)}{BC_i : (B\varphi, h(\tau))} \tag{8.1}$$

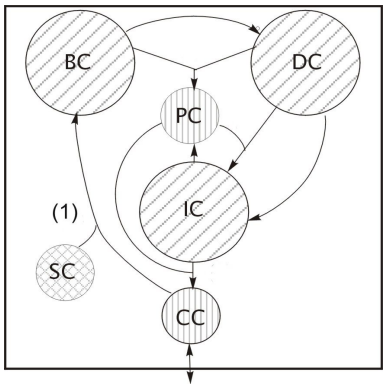

Figure 8.1: A g-BDI agent model including a SC for filtering information.

Figure 11.6 shows the extension of the graded BDI agent model with the different contexts (including the social context, SC) and some of the bridge rules relating them. Particularly, the bridge rule 8.1, in charge of the agent information filtering process, is illustrated as rule (1).

Example 4 *Suppose a tourism agent ($Agent_1$) is informed by another tourism agent ($Agent_2$) that "there is not enough snow in Mendoza for skiing"($\neg s$), represented by $N_{12}(\neg s)$. As they are competitive travel agents, the trust of $Agent_1$ in $Agent_2$ with respect to tourism issues is low, this may be represented in the SC_1 of the $Agent_1$ by the formula $(T_{12}(\neg s), 0.4)$. Then, the $Agent_1$ by using the bridge rule (see rule 8.1) and defining $h(\tau) = \tau$, she derives in her BC_1 the formula $(B_1(\neg s), 0.4)$ expressing that her belief degree on $\neg s$ is also low (at least 0.4).*[1]

8.3.2 Trust in Delegation

The aim at considering a Social Context in the g-BDI agent architecture is to model the social aspects of agency, particularly the trust in other agents. Previously a Social Context was introduced to filter the agent incoming information, taking into account the trust in the informant agents. The social trust must also be considered when the agent decides about the delegation to others of an action, part of the plan toward the goal. Thus, trust and delegation are closely related. In this subsection, we use the Social Context in order to represent the trust needed in order to evaluate the risk of delegating partial plans and thus decide whether or not to delegate them. We need to extend the SC defined above to represent and reason about this kind of trust. Different modalities are used to distinguish the diverse types. They are treated separately because $agent_i$ may trust $agent_j$ with respect to the information she gives about φ, but may not trust her about delegating an action related to φ.

[1] In the case that the $Agent_1$ has previous conflicting information about this subject, a belief revision process is needed to decide her current beliefs.

Language

Assuming we have a multiagent system with a finite set of agents: $\{agent_i\}_{i \in A_g}$, we extend the dynamic propositional language $\mathcal{L}_{\mathcal{D}}$ (see Section 3.1) used in BC to reason about actions and plans, by a family of modal operators T^d_{ij},[2] where $i, j \in A_g$.

Then, the language for the social context $\mathcal{L}_{\mathcal{SC}}$ is extended by the $\mathcal{L}_{\mathcal{D}}$-formulae. The modal formulae $T^d_{ij}\psi$ are included to represent the trust $agent_i$ has on $agent_j$ related to delegating an action in ψ, where ψ is a closed formula in $\mathcal{L}_{\mathcal{D}}$ (i.e. $\psi = [\alpha]\varphi$, $\alpha \in \Pi^0$ and $\varphi \in \mathcal{L}$). The formulae $T^d_{ij}\psi$ may be graded, taking values in [0,1], to express different levels of trust. Like in the other contexts, we use a many-valued approach for trust modelling. When who the $agent_i$ holding the trust is is clear from the context we remove its subindex, that is, $T^d_{ij}\varphi$ becomes $T^d_j\varphi$.

The theory for SC in a g-BDI agent, $\mathcal{T}_{\mathcal{SC}}$, will have closed formulae like $(T^d_j[\alpha]\varphi, \tau)$ expressing that the trust of the agent toward an $agent_j$ about a plan α directed to a goal φ, has degree greater than τ. We propose to use a plan classification based on an action ontology to assign the different levels of trust to dynamic formulae related to plans, preserving the following axioms:

$$T^d_j[\alpha \cup \beta]\varphi \equiv_L T^d_j[\alpha;\beta]\varphi \equiv_L T^d_j[\alpha]\varphi \wedge_L T^d_j[\beta]\varphi$$

Namely, the trust level in an $agent_j$ related to delegate her a plan that combines actions by concatenation or non-deterministic disjunction, results in the minimum of the trust levels with respect to delegate the elementary actions.

Actually, one in fact could assume that the trust only depends on the plan α and not in the goal the agent is trying to achieve with its execution. In such a case, the language could be simplified and instead of introducing the modal operators T^d_j one could introduce in the language a set of fresh many-valued propositional variables td^j_α indexed by agent identifiers $j \in A_g$ and by actions α from a given set of actions Π^0 that we may assume to be closed by nondeterministic union and concatenation. This approach would be similar to what was done with cost propositional variables C_α in the intention context IC (see Section 6.3). In that setting the above axioms should be replaced by analogous ones on the variables td^j_α:

$$td^j_{\alpha\cup\beta} \equiv_L td^j_{\alpha;\beta} \equiv_L td^j_\alpha \wedge_L td^j_\beta$$

Regarding intentions, we believe that in the case a g-BDI agent is evaluating an intention towards a goal φ by a plan α, when the plan execution involves the delegation of the plan (or subplan) to another $agent_j$, it needs to use trust. In previous works, as in [76], it was considered that the plan quality could be computed as a weighted sum of a *standard rating* (combination of the benefit obtained by the plan execution and its cost) and a *cooperative rating* (evaluated from the trust in the agents involved).

[2] We use the superscript d to differentiate the kind of trust (on delegation) we are dealing with.

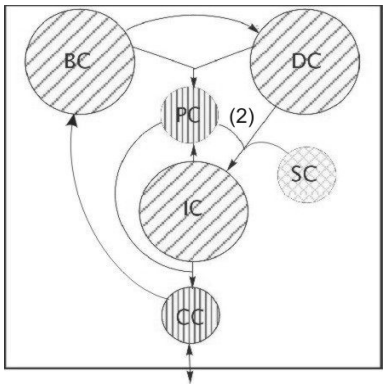

Figure 8.2: A g-BDI agent model including a SC to deal with delegation.

To compute the intention degrees in the g-BDI model, we have proposed a function that combines different factors (i.e. desire degree, belief in satisfying the goal, cost of the plan), that was formalized by defining an specific bridge rule (see 7.3 in Section 7.2). The intention degree is thought as a trade-off between the benefit of reaching a goal and the cost of the plan. Now, we propose to include the trust in the agent that will cooperate with the action execution. The following bridge rule computes the intention degree of $I_\alpha\varphi$ from the degree d of $D^+\varphi$, the degree r of belief $B[\alpha]\varphi$, the cost c of the plan α and the trust level t in the $agent_j$ that will cooperate in the execution of the plan α.[3]

$$\frac{DC : (D^+\varphi, d), BC : (B[\alpha]\varphi, r), PC : fplan(\varphi, \alpha, P, A, c), SC : (T_j[\alpha]\varphi, t)}{IC : (I_\alpha\varphi, f(d, r, c, t))} \quad (8.2)$$

Different functions f allow to model different agent behaviors. For instance, f may be defined as a weighted average,

$$f(d, r, c, t) = \frac{w_d.d + w_r.r + w_c.(1 - c) + w_t.t}{w_d + w_r + w_c + w_t}$$

where the different weights w_i are set according to the diverse agent types. For instance, for a *greedy agent* w_c will be set greater than the other weights and for a *suspicious agent* the more relevant will be w_t.

In Figure 8.2 the extended g-BDI agent model is illustrated, in this case we focus on the trust in delegation. The bridge rule 8.2 that computes the intention degrees taking trust into account is shown as rule(2).

Example 5 *The personal tourism agent (*T-Agent*) recommends tourism packages*

[3]If the trust in the $agent_j$ only depends on the plan α and not in the goal φ trying to achieve then, the premise $(T_j[\alpha]\varphi, t)$ may be replaced in the BR with (td^j_α, t).

*provided by two different tourism operators (*P1-Agent *and* P2-Agent*). It is important for the* T-Agent *to take into account the trust she has in the different Providers in the recommendation she undertakes. In this case, we consider that the trust depends only on the kind of tourist plan that the operator offers. For instance, we consider the region of the country as a classification element, since there are tour-operators that are good for plans in a particular region, but not in others.*

María, our tourist of Example 1 (see Section 6.2), activates the personal agent T-Agent*, to get an adequate plan, i.e. a tourist package, that satisfies her preferences (e.g. going to a mountain place (*m*) and practicing rafting (*r*) in a beautiful Argentinian place). The principal steps of the* T-Agent *towards to give a suitable recommendation are resumed next (for details see Section 7.4):*

- The T-Agent *finds that the plans* Mendoza *(*Me*) and* SanRafael *(*Sr*) are feasible plans for the combined goal* $m \wedge r$*, while* Cumbrecita *(*Cu*) is feasible only for* m*. The Planner also computes the normalized cost (*$c \in [0, 1]$*) of these plans being respectively:* $c_{Me} = 0.60$ *and* $c_{Sr} = 0.70$ *and* $c_{Cu} = 0.55$*.*

- In the belief context of the T-Agent *the following beliefs of achieving the different desires through the plans offered are computed as follows:* $(B[Me]m, 0.7)$*,* $(B[Me]r, 0.6)$*,* $(B[Me](m \wedge r), 0.6)$*,* $(B[Sr]m, 0.5)$*,* $(B[Sr]r, 0.6)$*,* $(B[Sr](m \wedge r), 0.5)$ *and* $(B[Cu]m, 0.4)$*.*

- The packages Mendoza *and* Cumbrecita *are provided by* P1-Agent *while* San-Rafael *is offered by* P2-Agent*. The trust* T-Agent *has on* P1-Agent *with respect to both packages* Mendoza *and* Cumbrecita*, is medium and the trust in* P2-Agent *providing* SanRafael *is higher. In* T-Agent *SC theory* $\mathcal{T}_{\mathcal{SC}}$ *the trust in these provider agents are represented using a set of many-valued propositional variables as the following formulae:* $(td^1_{[Me]}, 0.7)$*,* $(td^1_{[Cu]}, 0.7)$ *and* $(td^2_{[Sr]}, 0.9)$*.*

- Using bridge rule 8.2 with the weighted function proposed, the T-Agent *computes the different intention degrees towards satisfying the user preferences by following feasible plans. We consider that María has selected the* confidence *priority criterion and thus, the* T-Agent *adopts a suspicious behaviour by setting the following weights:* $w_d = w_r = w_c = 0.5$ *and* $w_t = 1$*.*

As a result of the previous process, the intention context contains the following formulae:

$(I_{Me}(m \wedge r), 0.685)$, $(I_{Sr}(m \wedge r), 0.735)$,
$(I_{Me}(m), 0.64)$, $(I_{Me}(r), 0.58)$,
$(I_{Sr}(m), 0.69)$, $(I_{Sr}(r), 0.63)$,
$(I_{Cu}(m), 0.655)$.

Then, the T-Agent *recommends María the plan* SanRafael *(*Sr*) since it brings the best cost/benefit relation (represented by the intention degree 0.735) respecting her*

priority selection (i.e. confidence). If we compare this result with the one obtained in the Example of Section 7.4 where the recommended package was Mendoza*, we can notice that the trust in the different package providers makes the difference in the recommendation.*

8.3.3 Conclusions

In this Chapter we have shown how the g-BDI agent model can be extended to include some social aspects.

We have first proposed how the language and semantics defined for the mental contexts could be extended to reason about the agent's different attitudes or the attitudes of other agents. In this way, we partially overcome the limitation of the previous languages that did not allow for nested modal formulae. Second, we have shown that the g-BDI agent model can be extended to deal with different kinds of trust, necessary for an agent who interacts in an agent society. With this aim, a social context has been added to the g-BDI agent model to represent different kinds of trust in other agents. Particularly, we have considered the trust in informant agents and the trust to delegate to other agents the execution of a plan (or partial plan). Preliminary results on the inclusion of a social context in the agent model can be seen in [31, 32]. Further research in this direction is ongoing and we consider very promising the recent work of Pinyol and Sabater-Mir in [120] focussing the integration of a cognitive model of reputation within a BDI agent architecture.

Design is not just what it looks like and feels like. Design is how it works.

S. Jobs

Chapter 9

Operational semantics for g-BDI Agents

9.1 Introduction

As exposed in Chapter 4 the graded BDI model of agents is based on multi-context systems. These systems are basically deductive machines. In this Chapter, we want to specify the operational semantics of this agent model.

Operational semantics is a way to give meaning to computer programs in a mathematically rigorous way. The semantics for a g-BDI model of agent will describe how a valid agent model is interpreted as sequences of computational steps. These sequences then are the meaning of the model. Operational semantics may define an abstract machine and give meaning to language expressions by describing the transitions they induce on a finite state machine. Alternatively, via a pertinent process calculus, operational semantics can be defined as syntactic transformations on sentences themselves. We decided to follow this second approach.

The process calculus approach has already been used to cope with formal aspects of multi-agent interactions defining different protocols [55]. Next, we outline some of these calculus.

The π-calculus is a process calculus developed by Milner et al. [106] as a continuation of the body of work on the process calculus CCS (Calculus of Communicating Systems) [107]. The aim of the π-calculus is to be able to describe concurrent computations whose configuration may change during the computation.

The Ambient Calculus due to Cardelli et al. [28] was developed as a way to describe the movement of processes (agents) and devices, including movement through boundaries (administrative domains). It can also be considered as an extension of the π-calculus and it is presented in more detail in next Section 9.2.

The Lightweight Coordination Calculus (LCC) [130] can also be considered as a variant of the π-calculus with an asynchronous semantics to coordinate processes that may individually be in different environments. LCC was designed specifically to formalize agent protocols for coordination and it is suitable to express interac-

tions within multi-agent systems without any central control. LCC borrows the notion of role from agent systems but reinterpret this in a process calculus. Social norms in LCC are expressed as the message passing behaviours associated with roles. The most basic behaviours are the send or receive messages, where sending message may be conditional on satisfying a constraint and receiving a message may imply constraints on the agent accepting it. The constraints are expressed by structured terms (i.e. Prolog syntax). More complex behaviours are specified using the connectives *then*, *or* and *par* for sequence, choice and parallelism, respectively. A set of such behavioural clauses specifies the message passing behaviour expected of a social norm. It is also possible for LCC to verify the protocols using automated means, e.g. model checking [150]. Walton in [148] presents a language based on CCS [107] to specify agent protocols in a flexible manner during the interaction of agents. Then, in [149] he proposes a Multi-agent Protocol (MAP) based in LCC and oriented to agent dialogues. These protocols allow to separate agent dialogues from their specific agent reasoning technology.

Ambient LCC [90] is a language based on process algebra concepts that combines the notions of LCC and ambient calculus. It was specially designed to support the execution of electronic institutions, an organization model for Multi-Agent Systems.

In order to give semantics to a g-BDI agent, we take advantage of Ambient calculus. Although process calculi have been mainly used to model multiagent systems, we have considered that the modular structure that Multi-context system (MCS) provide to the architecture of an agent would permit a similar treatment to single agents as well. Particularly we find that the notion of ambient is also suitable to represent the MCS main components: contexts and bridge rules.

As in Ambient LCC we combine Ambient calculus with some LCC elements but in this case, for dealing with the internal structure of intentional agents. We focus on the work about Ambient Calculus due to Cardelli et al. [28] to capture the notion of bounded ambient and we take into account some elements of LCC syntax [130] to represent the state components (e.g. terms, variables).

Since the g-BDI agent model is specified using multi-context systems (MCS), we first introduce a specific ambient calculus which we call Multi-context Calculus (MCC) with its corresponding semantics. The calculus presented is general enough to support the execution of different kinds of MCSs and particularly, we show how a graded BDI agent can be mapped to the calculus proposed.

9.2 Mobile Ambient Calculus

Ambient calculus was developed as a way to express mobile computation [28]. It can also be viewed as an extension of the basic operators of the π-calculus [106]. The inspiration behind Ambient calculus is the observation that many aspects of mobility involve administrative considerations. For example, the authorization to enter or exit a domain, and the permission to execute code in a particular domain. These

issues were principally motivated by the needs of mobile devices. However, they are very similar to the issues faced by agents in an open environment. The ambient calculus addresses this problem by defining an ambient (informally) as a "bounded space where computation happens". The existence of a boundary determines what is inside and outside the ambient. Process mobility is represented as crossing of boundaries and security is represented as the ability or inability to cross them. In turn, interaction between processes happens in shared locations within a common boundary. Ambients can also be nested, leading to a hierarchy. An ambient is also something that can be moved. For example, to represent a computer or agent moving from one place to another.

More precisely, each ambient has a name, a collection of local processes that run directly within the ambient, and a collection of sub-ambients. The syntactic categories are processes and capabilities. A process is analogous to an individual agent. A process may be placed inside an ambient, may be replicated, and may be composed in parallel with another process, which means that the processes execute together. In Ambient calculus, $n[P]$ denotes an ambient named n containing the process P. The formal syntax of Ambient calculus is shown in Table 9.1.

P, Q, R	::=	0	Inactivity
		$(\nu n)\,.P$	Restriction
		$P \mid Q$	Parallel Composition
		$M\,[P]$	Ambient
		$!P$	Replication
		$M.P$	Capability Action
M	::=	n	Name
		in M	can enter into M
		out M	can exit out of M
		open M	can open M
		ϵ	null
		$M.M'$	composite

Table 9.1: Syntax of Ambient calculus

In general, an ambient exhibits a tree structure induced by the nesting of ambient brackets. Each node of this tree structure may contain a collection of (non-ambient) processes running in parallel, in addition to subambients. We say that these processes are running in the ambient, in contrast to the ones running in subambients. The general shape of an ambient is, therefore:

$$n\,[P_1 \mid \cdots \mid P_k \mid m_1\,[...] \mid \cdots \mid m_j\,[...]]$$

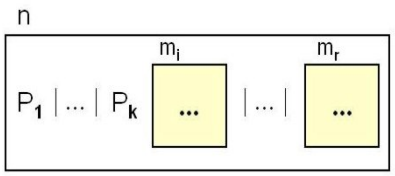

Figure 9.1: General structure of an ambient

To illustrate this structure we may display ambient brackets as boxes. Then the general shape of an ambient is shown in Figure 9.1.

One of the relevant characteristics of the ambient calculus is the definition of capabilities M for processes, which are described by actions. These capabilities permit things to happen within ambients. Especially, this calculus presents some actions related to crossing or opening ambient boundaries. Thus, different capabilities are defined as for example:

- Entering an ambient (*in* m capability): this action is used by a process to enter an ambient, i.e. to cross its boundary. The result is that the process (and its enclosing ambient) move from the current ambient to the ambient pointed in the action.

$$n\,[in\ m \cdot P|Q]\ \mid m\,[R] \rightarrow m\,[n\,[P|Q]\ \mid R]$$

- Exiting an ambient (*out* m capability): this action is used by a process to exit an ambient. The result is that the process (and its enclosing ambient) move outside the current ambient to a parent ambient according to the ambient hierarchy.

$$m\,[n\,[out\ m \cdot P|Q]\ \mid R] \rightarrow n\,[P|Q]\ \mid m\,[R]$$

For further information on the formal definition of Ambient calculus the reader is referred to [28]. Synthesizing, we can say that the emphasis of the Ambient calculus is on boundaries and their effect on computation, having the following key features:

- Ambients are used to separate locations and allow a hierarchical structure (defining a topology of boundaries).

- Process mobility is represented as crossing of boundaries, by the movement of processes between ambients.

- Security is represented as the ability or inability of a process to cross boundaries.

- Interaction between processes is by shared location within a common boundary (i.e. process can communicate only within the same ambient).

After these considerations, we find that the notion of ambient is also appropriate to represent contexts in Multi-context systems. Contexts encapsulate the local aspects of particular logical deductions in a global system and bridge rules enable to represent the interaction or compatibility between them. Then, each unit can be mapped to an adequate ambient having a state and a process running in it. Moreover, bridge rules may be also represented by special ambients whose mobile processes may be in charge of the inter-context deduction.

9.3 Multi-context Calculus

Multi-context systems (MCS) are specifications of deductive machines that modify the internal states of the different contexts through the context inner deductions and bridge rules [68]. In order to translate these MCS specifications into computable languages, we propose a Multi-context calculus (MCC) based on Ambient calculus. The notion of ambient allows us to encapsulate the states and processes of the different contexts and bridge rules. The possibility of structuring ambients hierarchically enables us to represent complex contexts where different components may be represented by different ambients.

We also take advantage of the process mobility addressed in Ambient calculus to represent the process attached to a bridge rule. This process is meant to supervise a number of context ambients to verify if particular formulae are satisfied and, if it is the case, to add a formula in another context ambient. Thus, this process will be getting in and out of the different ambients. Our definition of the actions for entering and exiting an ambient (i.e. *in* C and *out* C) is slightly different from the one used in Ambient calculus. In Ambient calculus a process gets into or out of an ambient C with the ambient enclosing it. In MCC calculus we want the process to move alone, then we redefine these capabilities by the following reduction rules that give semantics to *in* C and *out* C actions:

$$n\,[in\ m.P \parallel m\,[Q]] \ \rightarrow\ n\,[m\,[P \parallel Q]]^1$$

$$m\,[n\,[out\ n.P \parallel R]] \ \rightarrow\ m\,[P \parallel n\,[R]]$$

Furthermore, for defining our calculus we use some elements of LCC [130] as the definition of structured terms (i.e. words in a suitable language) as the necessary elements to represent the ambient state. In LCC terms are used to specify constraints that restrict the interchange of messages and to represent some post-conditions after the message sending. In our calculus, the ambient state formulae

[1] In MCC we use $\parallel$ to represent parallel composition instead of the symbol $|$, normally used in Ambient Calculus, as to differentiate parallelism from the choice symbol in BNF grammars.

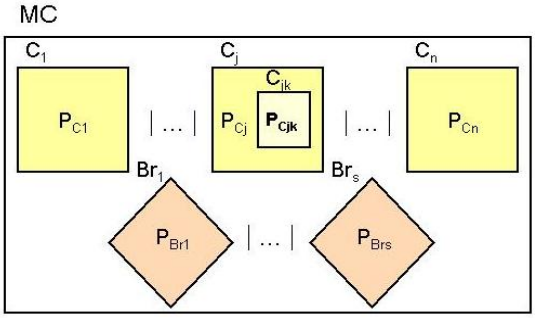

Figure 9.2: The general ambient structure in MCC

determine the results of the execution of the context ambient process (inner context deduction) and also can trigger some bridge rule processes (inter-context deduction).

In a Multi-context calculus (MCC), a MCS is structured by a global ambient, having an identifier and a *Clause* inside it. This clause may generate a set of clauses (ambients) for representing contexts and bridge rules. A context ambient has an identifier, a state and the context process being executed in it. Moreover, a context ambient may have other context ambients inside it, thus composing a nested structure of ambients. Besides, a bridge rule ambient has an internal state and a special process representing the inter-context deduction, attached to it. Such an ambient structure is illustrated in Figure 9.2.

The definition of the MCC syntax is shown in Table 9.2. In the following we describe the main syntax categories in the definition.

Multi-context System (MCS): is defined by an ambient structure where the global ambient identifier is Id_{MC} and *Clause* will result in the ambients and processes inside it (see (1) in Table 9.2). *Clause* leads us to a set of two type of clauses: $Clause_c$ and $Clause_{br}$ (2). $Clause_c$ generates a context ambient structure (possibly nested) with a context process P_c running in each ambient C (3). Respectively, $Clause_{br}$ becomes a bridge rule ambient Br (4) where a P_{br} process is being executed. In this way, we define a global ambient where different processes (P_c and P_{br} types) are running in parallel. As the processes are being executed in different ambients there is no possible interaction between them (i.e. interaction between processes happens only in a shared ambient).

Context ambient: this ambient has a context process running in it. The context ambient C is defined as $c(Id_c, S_c)$ where Id_c is its identifier and S_c its state (5). In turn, the state S_c is a set of $Terms$ of an adequate language $\mathcal{L}^c$ (e.g. Prolog formulae) that represents valid formulae in the context (7). In many

$$
\begin{array}{lll}
MCS & ::= & Id_{MC}\ [Clause] \quad (1)\\
Clause & ::= & (Clause_c \parallel Clause) \mid (Clause_{br} \parallel Clause) \mid \epsilon \quad (2)\\
Clause_c & ::= & C\ [P_c \parallel Clause_c] \parallel Clause_c \mid \epsilon \quad (3)\\
Clause_{br} & ::= & Br\ [P_{br}] \quad (4)\\
C & ::= & c(Id_c,\ S_c) \quad (5)\\
Br & ::= & br(Id_{br},\ S_{br}) \quad (6)\\
S_c & ::= & \{Term\} \quad (7)\\
S_{br} & ::= & L \quad (8)\\
L & ::= & \langle U \rangle \quad (9)\\
U & ::= & \langle V \rightarrow Term \rangle \quad (10)\\
V & ::= & variable \quad (11)\\
P_c & ::= & Clause_c \mid \vdash_c \mid P_c \cdot P_c \mid P_c\ or\ P_c \mid \\
 & & if\ Term\ then\ P_c\ else\ P_c \mid Action \quad (12)\\
Action & ::= & in\ C \mid out\ C \mid get^*(Term, L) \mid getS(L_1, ..., L_n, L) \mid \\
 & & add^*(L, Term) \mid remove(C, Term) \mid \epsilon \quad (13)\\
P_{br} & ::= & Clause_{br} \mid (spy(Br, C_1, \varphi_1, L_1) \parallel spy(Br, C_2, \varphi_2, L_2) \parallel \ldots \\
 & & \parallel spy(Br, C_n, \varphi_n, L_n)) \cdot put^*(Br, C_k, \varphi_k, L_1, ..., L_n) \quad (14)
\end{array}
$$

$$
spy(Br, C, Term, L) ::= out\ Br \cdot in\ C \cdot get^*(Term, L) \cdot out\ C \cdot in\ Br \quad (15)
$$

$$
\begin{array}{lll}
put^*(Br, C, Term, L_1, ..., L_n) & ::= & out\ Br \cdot in\ C \cdot getS(L_1, ..., L_n, L) \cdot \\
 & & add^*(L, Term) \cdot\ revise(C) \cdot\ out\ C \cdot in\ Br \quad (16)
\end{array}
$$

Table 9.2: Syntax of Multi-context calculus (MCC)

cases it may be useful to use a nested structure of context ambients. For example, to represent a complex context where its language or deduction system are built using different layers. In a nested structure of ambients we can deal with this complexity defining different ambients for each layer. In the MCC syntax it is possible to represent a context ambient structure: from $Clause_c$ we can generate parallel context ambients (at the same level of hierarchy) or embedded context ambients, by using the rewriting rule (3).

Context process: consists of a deductive operator $\vdash_c$ corresponding to the context logical deduction. The P_c may be composed using the basic operators: sequential processing (.); deterministic choice (*or*) meaning that if at all possible the process on the left is to be executed, otherwise, the right one is chosen; and

the classical conditional $if\ then\ else$. Furthermore, rewriting P_c as $Clause_c$ the recursion of processes is allowed. Then, different kinds of programs may be represented by P_c (12).

Bridge rule ambient: this ambient has a special process P_{br} running in it (4). These ambients are defined as $br(Id_{br}, S_{br})$, having an identifier Id_{br} and a state S_{br} (6). The state for a Br ambient is a kind of substitution memory L composed by the substitution lists returned by the P_{br} process (8).

Bridge rule process: this process is a key characteristic in the MCC and represents the inter-context deduction process of a certain bridge rule (14). Each P_{br} is composed by a finite set of parallel $spy(br, C, Term, L)$ processes followed by a $put^*(Br, C, Term, L_1, ..., L_n)$ process. In the following items we describe in some detail these important components:

- $spy(Br, C, Term, L)$ process (15) gets out of the Br ambient and gets into the C ambient. In this ambient it retrieves in L all the substitution lists that result of unifying $Term$ with formulae in the context state. This task is done by the process $get^*(Term, L)$, which is the heart of the *spy* process. Then, it returns to the Br ambient.

- $put^*(Br, C, Term, L_1, ..., L_n)$ process (16) is executed after all the lists of substitutions L_1,...,L_n have been extracted by the different processes $spy(Br, C_i, Term_i, L_i)$, $i = 1, ..., n$. This process gets out of the Br ambient, comes into the C ambient and using the $getS(L_1, ..., L_n, L)$ process, retrieves in L all the substitutions compatible with the lists of substitutions L_1,...,L_n. Then, using the $add^*(L, Term)$ process, adds all the instances of $Term$ applying the resulting substitutions in L. In order to maintain the consistency in the ambient state, as the $add^*(L, Term)$ process may introduce new formulae in it, a $revise(C)$ process is needed.

- $revise(C)$ process is defined according to a suitable revision method chosen to keep the ambient state consistent. If we want to revise using time considerations as for example, allowing in the state to retain the more recent formulae respect to the conflicting ones, the insertion time t of a formulae in an ambient state, must be included in the calculus. In our case that means that each $Term$ in the context ambient state S_c needs a parameter t that will only be used by the revise process. Since in some revision processes we may need to remove formulae from the state, we include the $remove(C, Term)$ as a possible action.

9.4 Operational Semantics

One of the purposes of defining the MCC is to provide the Multi-context computational model with a clean and unambiguous semantics, allowing to be interpreted in a consistent way. There are different methods for giving semantics to a process calculus as for example, defining structural congruence between processes and reduction relations [28], or using rewriting rules for the clause expansion [130]. We have chosen the *natural semantics* methods to provide operational semantics for the MCC. This formalism is so called because the evaluation rules are in some way similar to natural deduction and it has been used to specified the semantics of Multi-Agent Protocols (MAP) [149]. In natural deduction we define relations between the initial and final states of program fragments. Thus, we found it suitable for our case since each process may produce some changes in the ambient state. A program fragment in our model is either a context process P_c or a bridge rule process P_{br}.

We define the evaluation rules for the different processes. The general form of these rules is: $M, a \diamond P \Rightarrow \ M'$, where M is the MCS at the start of the evaluation, a is the ambient (C or Br type) where the procedure P is executed and M' is the final global system.

I- Evaluation rules for context processes: $M, C \ \diamond \ P_c \Rightarrow \ M, C'$

Since each context process P_c runs in a particular context C of M and its execution only changes its state, in the following evaluation rules we can omit the reference to M. As the context ambient C is defined as $c(Id_c, S_c)$, we represent as C' the modification of its ambient state i.e. $C' = c(Id_c, S'_c)$.

$$C \diamond (\vdash_c) \Rightarrow \ C' \tag{9.1}$$

$$\frac{C \diamond P_{c1} \Rightarrow \ C' \quad C' \diamond P_{c2} \Rightarrow \ C''}{C \diamond P_{c1} \cdot P_{c2} \Rightarrow \ C''} \tag{9.2}$$

$$\frac{C \diamond P_{c1} \Rightarrow \ C'}{C \diamond P_{c1} \ or \ P_{c2} \Rightarrow \ C'} \tag{9.3}$$

$$\frac{C \diamond P_{c1} \Rightarrow \ C \quad C \diamond P_{c2} \Rightarrow \ C''}{C \diamond P_{c1} \ or \ P_{c2} \Rightarrow \ C''} \tag{9.4}$$

$$\frac{C \vdash Term \quad C \diamond P_{c1} \Rightarrow \ C'}{C \diamond \ if \ Term \ then \ P_{c1} \ else \ P_{c2} \Rightarrow \ C'} \tag{9.5}$$

Notice that $C \vdash Term$ represents that $Term$ is a valid formula in the ambient

state S_c, i.e. $Term \in S_c$.

$$\frac{\begin{array}{l} C \vdash Term \\ C \diamond P_{c2} \Rightarrow\ C'' \end{array}}{C \diamond\ if\ Term\ then\ P_{c1}\ else\ P_{c2} \Rightarrow\ C''} \tag{9.6}$$

II- Evaluation rule for bridge rule process: $M, Br\ \diamond\ P_{br} \Rightarrow\ M'$

As the fundamental processes for the P_{br} definition are the processes $get^*(Term, L)$, $getS(L_1, ..., L_n, L)$ and $add^*(L, Term)$, defining their semantics is enough to have the complete P_{br} semantics well defined. In some rules we use $\emptyset$ to denote that the result of the process execution is independent of the ambient where it is running.

$$\frac{\forall\ Term_i\ \left\{ \left\{ \begin{array}{l} C \vdash Term_i \\ \emptyset\ \diamond unify(Term, Term_i) \Rightarrow\ U_i \end{array} \right\} \leftrightarrow member(U_i, L') \right\}}{\emptyset\ \diamond get^*(Term, L) \Rightarrow\ L = L'} \tag{9.7}$$

Intuitively, $get^*(Term, L)$ gathers in the list L all the substitutions U_i that result from unifying $Term$ with $Term_i$, formulae in C ambient state.

$$\frac{\begin{array}{l} \forall\ (U_1 \in L_1, \ldots, U_n \in L_n) \\ \{(\emptyset\ \diamond unify^*(U_1, \ldots, U_n) \Rightarrow\ L^*) \leftrightarrow member(L^*, L')\} \end{array}}{\emptyset\ \diamond\ getS(L_1, \ldots, L_n, L) \Rightarrow\ L = L'} \tag{9.8}$$

Where $unify^*(U_1, \ldots, U_n)$ is a variant of the classical $unify$ function, where lists of substitutions $(U_1, \ldots, U_n)$ instead of formulae are unified. If $unify^*$ succeeds, its result is a list L^* of the unified substitutions.

$$\frac{\begin{array}{l} C = c(Id_c, S_c) \\ \forall L_i \{\ member(L_i, L) \leftrightarrow (Term\,[L_i] \in TermSet)\} \end{array}}{C \diamond\ add^*(L, Term) \Rightarrow\ c(Id_c, S_c \cup TermSet)} \tag{9.9}$$

Intuitively, $add^*(L, Term)$ adds to the ambient state S_c all the instances of the $Term$ formula by applying the substitutions in L.

9.5 Mapping a g-BDI Agent to the MCC

Given a g-BDI agent defined by its multi-context specification (see Chapter 4):

$$A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br})$$

we want to map it into the MCC language. Thus, we need to define a mapping $\mathcal{F} : \{A_g\} \mapsto MCC$, which maps each g-BDI agent A_g with its multi-context components (contexts and bridge rules) to the MCC language. The general insights of the mapping $\mathcal{F}$ between these two formalisms are the following:

Global ambient: the multi-context agent A_g is mapped to a global ambient A_g in MCC:

$$\mathcal{F} : \ A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br}) \ \mapsto \ A_g \, [Clause]$$

Context ambient: each context $C_i \in \{BC, DC, IC, PC, CC\}$ in the agent A_g, either mental or functional, is mapped to a suitable ambient structure (possibly nested) in MCC.

$$\mathcal{F} : \ C_i = \langle \mathcal{L}_i, A_i, \Delta_i, T_i \rangle \ \mapsto \ c(C_i, S_{C_i}) \, [P_{C_i} \parallel C_{i0} \, [P_{C_{i0}} \parallel [C_{i1} \parallel ...]] \,]$$

- **Language:** before setting the ambient state for a context C_i, we have to define the ambient language $\mathcal{AL}^{C_i}$. Since the languages of different mental contexts in the g-BDI agent model are built using different language layers, we create the corresponding ambient hierarchical structure where the inner an ambient is, the more basic language it has. The ambient state will be composed by formulae of the top level language. This structure allows us to differentiate the language layers (represented by $\mathcal{AL}^{C_{ij}}$) in different ambient states and by using the mobility of processes we can access the different formulae in them.
  $\mathcal{F} : \mathcal{L}_i \ \mapsto \ \left\{ \mathcal{AL}^{C_i}, \mathcal{AL}^{C_{i0}}, ..., \mathcal{AL}^{C_{ik}} \right\}$

- **Context ambient state:** the initial ambient state S_{C_i} is composed by the translation of the theory T_i formulae into the ambient language.
  $\mathcal{F} : T_i \ \mapsto \ S_{C_i} \subset \mathcal{AL}^{C_i}$

- **Context ambient process:** the process P_{C_i} attached to a context ambient is derived from its logical deduction system. Thus, it is built from the context theory, axioms and inference rules.
  $\mathcal{F} : \langle A_i, \Delta_i, T_i \rangle \ \mapsto \ P_{C_i}$

Essentially the P_{C_i} process is composed by the following sequential schema: $P_{C_i} ::= P^*_{A_i} \cdot P^*_{\Delta_i}$, where the $P^*_{A_i}$ process represents the generation of finitely-many instances of the different context axioms i.e. $P^*_{A_i} ::= P^*_{A_{i1}} \cdot ... \cdot P^*_{A_{in}}$, where the A_{ij}'s are axioms in A_i. Respectively, $P^*_{\Delta_i}$ is composed by processes in charge of generating the instances of the different inference rules. i.e. $P^*_{\Delta_i} ::= P^*_{\Delta_{i1}} \cdot ... \cdot P^*_{\Delta_{ik}}$, where the Δ_{ij}'s are rules in Δ_i. These processes are described in more detail for DC context in next Subsection 9.5.1.

Bridge rule ambient: each bridge rule Br_i is mapped to a suitable ambient Br_i having as internal state a list of possible substitutions L_i and a special process P_{Br_i}. The definition of both elements related to the Br_i ambient (i.e. L_i and P_{Br_i}) depends on the premise and conclusion of the bridge rule that it represents:

$$\mathcal{F}: \; Br_i = \frac{C_1 : \varphi_1, \ldots, C_n : \varphi_n}{C_k : \varphi_k} \;\mapsto\; br(Br_i, L_i)\,[P_{Br_i}]$$

- **Internal state:** is the list L_i of n substitution lists, i.e.: $L_i = \langle L_{i1}, \ldots, L_{in} \rangle$ where each sublist L_{ij} will contain the resulting substitutions of unifying the formulae φ_j with formulae in the context C_j.

- **Bridge rule process:** the special process P_{Br} is created in MCC (see (12) in Table 9.2) to represent the bridge rule inference. This process will add instances of formula φ_k in ambient C_k when the preconditions are satisfied.

$$P_{Br_i} ::= (spy(Br, C_1, \varphi_1, L_1) \parallel \ldots \parallel spy(Br, C_n, \varphi_2, L_n)) \cdot \\ \cdot put^*(Br, C_k, \varphi_k, L_1, ..., L_n)$$

The ambient structure in MCC for representing a g-BDI agent A_g is thus illustrated in Figure 9.3.

In summary, for each mental or functional context in the g-BDI agent specification, we can define the corresponding ambient structure in MCC. Since the planning and communication contexts are based in first order logic, the mapping is straightforward and both contexts can be easily mapped in an ambient. Namely, both ambient languages has only one layer, the theories may be translated to the initial context states and the inference rule resolution may be translated to the corresponding ambient processes.

In the case of the mental contexts, since the logical framework is more complex, some details must be analyzed. As a matter of example, in the next subsection we describe the mapping $\mathcal{F}$ for the Desire Context. In a similar way, the ambients for the other mental contexts are defined.

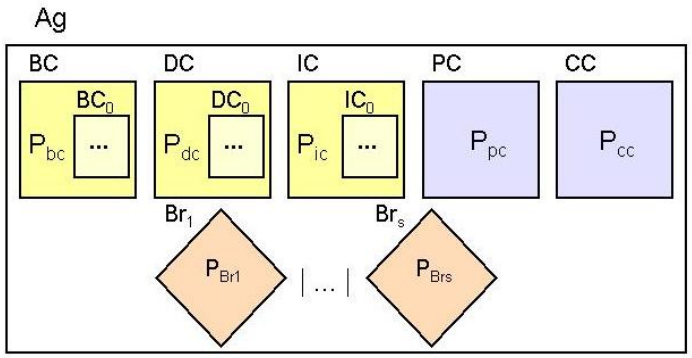

Figure 9.3: The ambient structure for a g-BDI agent

9.5.1 Mapping the Desire context into a Desire ambient

We want to define a mapping from the desire context DC to a suitable ambient structure in MCC.

$$\mathcal{F} : DC = \langle \mathcal{L}_{DC}, A_{DC}, \Delta_{DC}, T_{DC} \rangle \ \rightarrow \ c(DC, S_{DC}) \, [P_{DC} \parallel ...]$$

For this, we start with an abridged description of the components of the DC context: the language $\mathcal{L}_{DC}$, the axioms A_{DC}, the inference rules Δ_{DC} and a theory T_{DC}. A more complete description can be found in Chapter 6.

Language ($\mathcal{L}_{DC}$): it is defined over a (classical) propositional language $\mathcal{L}$ (generated from a finite set of propositional variables and connectives $\neg$ and $\rightarrow$) by introducing two (fuzzy) modal operators D^+ and D^-. As in other mental contexts, we use a (modal) many-valued logic to formalize reasoning about graded desires by interpreting the (positive and negative) degrees of desires over a (classical) proposition φ as the truth-degrees of the modal formulas $D^+\varphi$ and $D^-\varphi$ respectively. We choose Łukasiewicz logic, extended with rational truth-constants, as the underlying many-valued logic dealing with the many-valued modal formulas. The $\mathcal{L}_{DC}$ language is built therefore as follows:

- If $\varphi \in \mathcal{L}$ then $D^-\varphi, D^+\varphi \in \mathcal{L}_{DC}$
- If $r \in \mathbb{Q} \cap [0,1]$ then $\overline{r} \in \mathcal{L}_{DC}$
- If $\Phi, \Psi \in \mathcal{L}_{DC}$ then $\Phi \rightarrow_L \Psi \in \mathcal{L}_{DC}$ and $\neg_L \Phi \in \mathcal{L}_{DC}$ (where $\neg_L$ and $\rightarrow_L$ correspond to the negation and implication of Łukasiewicz logic, other logic connectives, like $\wedge_L, \vee_L, \equiv_L$ are definable from $\neg_L$ and $\rightarrow_L$)

Axioms and inference rules (A_{DC} and Δ_{DC}): to axiomatize the logical system DC we need to combine different sets of axioms Axioms:

(CPC) Axioms of classical logic for non-modal formulas

(RPL) Axioms of Rational Pavelka logic for modal formulas

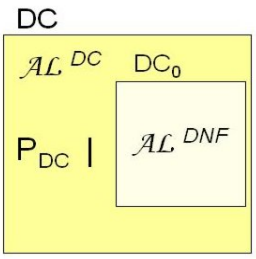

Figure 9.4: The DC ambient structure

(BD0$^+$) $D^+(A \vee B) \equiv_L D^+A \wedge_L D^+B$
(BD0$^-$) $D^-(A \vee B) \equiv_L D^-A \wedge_L D^-B$

Inference Rules:

(MP1) modus ponens for $\rightarrow$
(MP2) modus ponens for $\rightarrow_L$
Introduction of D^+ and D^- for implications:
(ID$^+$) from $\varphi \rightarrow \psi$ derive $D^+\psi \rightarrow_L D^+\varphi$
(ID$^-$) from $\varphi \rightarrow \psi$ derive $D^-\psi \rightarrow_L D^-\varphi$.

Theory (T_{DC}): is a set of $\mathcal{L}_{DC}$ formulae.

Now we are ready to define the corresponding Desire Ambient $\mathcal{F}(DC)$ by describing its state language $\mathcal{AL}^{DC}$, its initial state S_{DC} and its process P_{DC}.

Desire ambient

1. **State Language**

   Since the modal language $\mathcal{L}_{DC}$ for the desire context is built in two layers (one base propositional language $\mathcal{L}$ and the modal $\mathcal{L}_{DC}$), we define two ambients to represent these language layers. We define the ambient DC_0 (to represent language $\mathcal{L}$) inside the ambient DC (to represent language $\mathcal{L}_{DC}$), having the following ambient structure: $DC\ [P_{DC}|DC_0]$. This structure and the languages involved are illustrated in Figure 9.4.

   This nested ambient structure enable us to deal in a proper way with the different fomulae in the two language layers. The language for the DC_0 ambient is the basic language used in the DC context for building the language $\mathcal{L}_{DC}$.

   As it is convenient for the definition of the deductive process P_{DC}, we consider that the formulas of this language are in Disjunctive Normal Form (DNF). So we have for DC_0 the mapping $\mathcal{F} : \mathcal{L} \mapsto \mathcal{AL}^{DNF}$ defined by

- $\mathcal{F}(\psi) \;=\; \psi^{DNF}$

The mapping from the language $\mathcal{L}_{DC}$ to the language $\mathcal{AL}^{DC}$ for the desire ambient, $\mathcal{F} : \mathcal{L}_{DC} \;\mapsto\; \mathcal{AL}^{DC}$ is then defined as follows:

- $\mathcal{F}\ (D^{+}\varphi) \;=\; d^{+}(\mathcal{F}(\varphi))$
- $\mathcal{F}\ (D^{-}\varphi) \;=\; d^{-}(\mathcal{F}(\varphi))$
- $\mathcal{F}\ (\overline{r}) \;= r$
- $\mathcal{F}\ (\neg_{L}\Phi) \;=\; neg(\mathcal{F}(\Phi))$
- $\mathcal{F}\ (\Psi \rightarrow_{L} \Phi) \;=\; imp(\mathcal{F}(\Psi), \mathcal{F}(\Phi))$

2. **Initial state**

   The DC ambient state S_{DC} is composed by the translated formulae of the context theory:

   $S_{DC} \;=\; \{\mathcal{F}(\varphi) |\ \varphi \in T_{DC}\}$

3. **DC Process P_{DC}**

   We need to map the logical deduction of the desire context DC, composed by two different layers of axioms and inference rules, into the P_{DC} process. Actually, it can be shown that reasoning in the DC axiomatic system can be reduced to reasoning in plain Łukasiewicz logic from a big, but finite, theory which gathers suitable translations of instances of all the axioms and inference rules, and of the formulas of the context theory T_{DC}. We will consider deduction in Łukasiewicz logic as an encapsulated process $P_{\text{Ł}}$ without entering in its internals. This is possible to engineer by using one of the existing theorem provers for this many-valued logic (e.g. [7]). We describe next how to build such a theory in the context ambient which incorporates a finite set of instances of the axioms and inference rules that model the behavior of D^{+} and D^{-}. The idea is that, since we have a language $\mathcal{L}$ built over a finite set of propositional variables, there are only a finitely-many different DNF formulas, so there are finitely-many instances of axioms and rules over these DNF formulas. Therefore the P_{DC} process will consist of two parts. The first one, involving four processes $P_{V}, P_{DNF}, P_{AX}, P_{\Delta}$, will add to the initial ambient state S_{DC} (context theory) the set of instances of the axioms and inference rules, changing the initial state into S'_{DC}:

$$c(DC, S_{DC}) \diamond\ (P_{V}(VSet) \cdot P_{DNF}(VSet) \cdot P_{AX} \cdot P_{\Delta}) \Rightarrow\ c(DC, S'_{DC})$$

   Then, over the state S'_{DC}, the deduction over Łukasiewicz logic, represented by the process $P_{\text{Ł}}$ can be applied. Thus, the P_{DC} process is defined as the

following schema of sequential processes:

$$P_{DC} \ ::= \ P_V(VSet) \cdot P_{DNF}(VSet) \cdot P_{AX} \cdot P_{\Delta} \cdot P_{\mathrm{L}}$$

In the following items we describe the four first processes:

- The $P_V(VSet)$ process extracts from S_{DC} the finite set of propositional variables appearing in the formulas of T_{DC} and puts them in $VSet$.

- The $P_{DNF}(VSet)$ process enters in the DC_0 ambient and through the $add^*_{DNF}(VSet)$ process creates and adds to S_{DC_0} the finite set of DNF formulae built upon the variables in $Vset$, i.e.:

$$P_{DNF}(VSet) ::= in\ DC_0 \cdot add^*_{DNF}(VSet) \cdot out\ DC_0$$

- The P_{AX} process is composed by all the processes derived from each context axiom. For this particular case of the DC ambient we have:

$$P_{Ax} ::= P_{Ax_1} \cdot P_{Ax_2}$$

  These processes implement respectively the axioms Ax_1 and Ax_2 (see below), for instance P_{Ax_1} is defined as:

$$\begin{aligned} P_{Ax_1} \ ::= \ & in\ DC_0 \ \cdot \ get^*(dpair(x,y), L) \cdot out\ DC_0 \ \cdot \\ & \cdot \ add^*(\mathcal{F}(D^+(x) \wedge D^+(y) \equiv D^+(x \vee y)), L) \end{aligned}$$

  where the special component processes have the following meaning:
  - $get^*(dpair(x,y), L)$ stores in L all the pairs (x,y) satisfying the condition $dpair(x,y)$: $x, y \in S_{DC_0}$ and $x \neq y$;
  - $add^*(\mathcal{F}(D^+(x) \wedge D^+(y) \equiv D^+(x \vee y)), L)$, using the pairs $(x,y) \in L$ for substitution, instantiates and adds to the ambient state S_{DC} the formulae $\mathcal{F}(D^+(x) \wedge D^+(y) \equiv D^+(x \vee y))$.

  In a similar way the P_{Ax2} process implements the corresponding axiom for D^-.

- The P_{Δ} process is composed of the processes representing the instances of the different inference rules. For the DC ambient there are two processes representing the rules Δ_1 and Δ_2, hence

$$P_{\Delta} ::= P_{\Delta_1} \cdot P_{\Delta_2} \ , \qquad \text{with}$$

$$P_{\Delta_1} ::= in\ DC_0 \cdot get^*(\mathcal{F}(x \rightarrow y), L) \cdot out\ DC_0 \cdot add^*(\mathcal{F}(D^+(y) \rightarrow D^+(x)), L)$$

and similarly for P_{Δ_2}, where

  - $get^*(\mathcal{F}(x \rightarrow y), L)$ gets the pairs (x, y) resulting from the unification of $\mathcal{F}(x \rightarrow y)$ in DC_0 ambient;
  - $add^*(\mathcal{F}(D^+(y) \rightarrow D^+(x)), L)$ adds all the instances of the formula $\mathcal{F}(D^+(y) \rightarrow D^+(x))$ with pairs $(x, y) \in L$.

- The final process $P_{\text{Ł}}$ applies Łukasiewicz logic deduction $\vdash_{\text{Ł}}$ to the state S'_{DC} resulting from the previous processes, i.e. $P_{\text{Ł}} ::= \vdash_{\text{Ł}}$

9.6 Conclusions

In this Chapter we have defined a MCC calculus for Multi-context systems (MCS) execution. The MCC proposed is based on Ambient calculus [28] and includes some elements of LCC [148]. The operational semantics for this language was given using Natural Semantics. We expect that MCC will be able to specify different kinds of MCSs. Particularly, we have shown how graded BDI agents can be mapped to this calculus. Preliminary work on MCC calculus and its uses to define the g-BDI agent semantics, can be seen in [36].

Through MCC we give to this agent model computational meaning and in this way, we are getting closer to the development of an interpreter of the g-BDI agents. Although process calculi have been mainly used in the past to model multiagent systems, we have considered that the modular structure that MCS provide to the architecture of an agent would permit a similar treatment to single agents as well (or to any system with a self-similarity structure like Holons). We think that the implementation of agent architectures using process calculi, in particular ambient calculus, would give a uniform framework for agent architectures, multiagent systems and also electronic institutions.

Part III

Methodology and a Case-Study

Chapter 10

Case Study Domain: Tourism Recommender Systems

In this Chapter we explain the a case study where we have applied our g-BDI model of agent: a tourist recommender system.

10.1 Introduction

In the last years, the Artificial Intelligence (AI) community has carried out a great deal of work on recommender systems [127, 108]. This kind of systems can help people to find out what they want, especially on the Web. As a result, the idea of recommender systems has been widely accepted among users.

Agent technology becomes invaluable to model different characteristics we expect from these recommender systems. When this technology is applied we obtain a community of distributed, complex and autonomous recommender agents. These software agents can learn the interests of users and make recommendations accordingly. The agents learn by tracking the actions of the user or by seeking explicit feedback from him. The most common applications of recommender agents are Web content filtering systems and e-commerce shop assistants (e.g., amazon.com). These agents are used to discover a person's interests in the hope of providing useful information or encouraging a sale.

From within the recommender systems we have select the tourism domain. The travel and tourism industry is one of the most important and dynamic sectors in Business-to-consumer (B2C) e-Commerce. In this context, recommender applications can be valuable tools supporting, for example, information search, decision making, and package assembly.

Another reason to choose tourist domain for this case study is its richness and different characteristics where diverse user's preferences and restrictions can be considered. Because of this variety, the recommendation systems can be treated in different levels of complexity. Besides this, its vocabulary and concepts are well known by people and then, a tourist recommender system can be easily used and

validated by general users.

10.2 Recommender Agents

Recommender systems are the technical response to the fact that we frequently rely on other people's experience and recommendations when confronted with a new field of expertise, where we do not have a broad knowledge of all facts, or where such knowledge would exceed the amount of information humans can cognitively deal with.

The main task of a recommender system is to locate items (movies, music, books, news, web pages, etc.) related to the interest and preferences of a single person or a group of people. This involves the construction of user models and the ability to anticipate and predict user preferences. To do this the user's profile is compared to some reference characteristics. These characteristics may be from the information item (the content-based approach) or the user's social environment (the collaborative filtering approach). When building the user's profile a distinction is made between explicit and implicit forms of data collection.

By looking at definitions of the design space of recommender systems [127, 145], there seems to be some basic items common to most of them [113], their relations are illustrated in Figure 10.1:

Resources: The targets of the recommendation process.

Recommenders: Those entities that give out opinions about resources. In practice recommenders are actors and they could also be artificial agents.

Descriptions: Those information about resources that include opinions.

Preferences: Recommendation seeker's position towards resources.

Techniques for Computing Recommendation: The system's means for automatically evaluating resources by using descriptions and preference information.

Recommendations: The concrete results of the evaluation process for the recommendation seeker. The recommendations may be presented in different ways (e.g. by filtering out resources, ordering resources).

Many different recommender systems were developed since the 1990s. A detailed taxonomy of recommender agents, classified by the application domain and by the different task-achievement techniques used, is presented in [108]. The diverse approaches may be grouped in the following major system types [14, 145]:

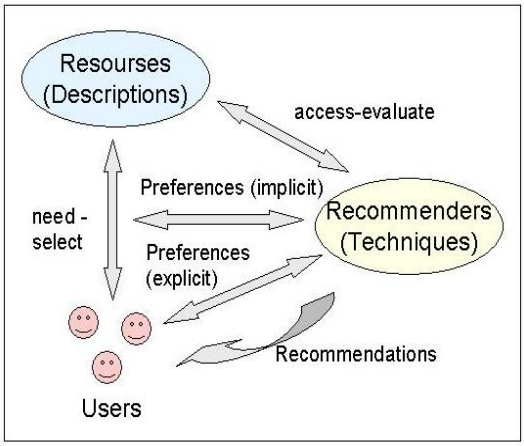

Figure 10.1: Principal items in a recommender schema

- **Collaborative Filtering (CF)**- These algorithms (also referred to as social filtering) focus on the behaviour of users on items, which are to be recommended, rather than on the internal nature of the items themselves. These systems concern with techniques for matching people based on their preferences and weighting the interests of people with similar taste, to produce a recommendation for the information seeker. This social approach, which most closely resembles the nature of real-life recommendations, is related to both the concept of collaborating individuals and the process of finding persons with similar interests.

- **Content-based Filtering (CBF)**- Focusses on the internal nature of items, or on the content of description files. These systems use two main classes of algorithms: information retrieval or attribute-based filtering algorithms. A content-based approach favors the semantics of the content over social interactions or user behaviour. Usually, CBF techniques use product descriptions (e.g. extracting a set of keywords), compute the users preferences (e.g. keywords which are contained in products selected by the user), and build the list of recommendations by searching for products that match the users preferences.

- **Knowledge-based Filtering** - These systems rely on an explicit representation of knowledge, usually as collections of statements, ontologies or other forms of rule systems. While the high performance and flexibility make the knowledge-based approach suitable for most tasks, applications with a strong focus on content or social semantics can be realized efficiently using the respective specialized approach (CBF or CF). If an application requires knowledge reasoning or inference, choosing the knowledge-based approach allows the developers to benefit from the existing components (e.g. knowledge representa-

tion and rule-based systems).

- **Social data mining systems** - Social data mining systems do not require users to engage in any new activity; rather, they seek to exploit user preference information implicit in records of existing social activity (e.g. Usenet messages, system usage history, citations or hyperlinks). The interactions between individuals are analyzed to understand innovation, collective decision making and problem solving, and how the structure of organizations and social networks impact these processes. Analysis of such inherently relational datasets is currently being applied in e-commerce to drive recommendation systems.

- **Hybrid Systems** - Can merge any combination of the above methods and metrics. Typically, hybrid recommender systems would compute ratings from a number of internal algorithms, before combining these in a single metric. In some cases, the preliminary results of the internal algorithms are stored component-wise in a vector, before crafting a single-dimensional rating for ranking.

Collaborative filtering (CF) technique is the most widespread, used in the earliest recommender systems [127]. In some application domains, the content of an item may be crucial. In these cases, recommender systems should use a content-based approach rather than a social approach (see e.g., [110, 105]). CF and CBF technologies exploit user preferences and allow acceptable recommendation accuracy for frequently bought (or selected) products such as music, video DVDs, books, Internet radio, etc. When accessing product descriptions, a list of CF recommendations is available for users in the section "Customers who bought this item also bought", while the CBF suggestions can be accessed via "Look for related items by keyword" and "Look for similar items by category" links.

There are other types of products which are less frequently bought and their purchase is related to higher risks (e.g., financial services, cars, electronic goods, services in the tourism domain). When recommending such products, recommender applications must support a more detailed elicitation of user requirements. Deep domain knowledge has to be exploited in order to be able to make more precise and more trusted recommendations. Knowledge-based (KB) recommender technologies [24, 58] are based on a detailed domain description in the form of structured product descriptions and constraints. The identification and construction of user preferences usually takes place in the context of an explicit sales dialog. The major advantage of this type of recommendation technology is the explicit representation of product, marketing and sales knowledge. Such a representation makes it possible the development of explanations argumenting why a certain product fits the wishes and needs of a given customer.

Agent technology becomes invaluable by appreciating the facts that we want the systems to take personal preferences into account, and to infer and intelligently

aggregate relationships from heterogeneous and distributed sources and data. Furthermore, we want systems to be scalable, open, privacy-protecting and we want to get the recommendations with the least possible work on users' behalf. Multi-agent systems permit the creation of subcommunities of agents for recommending among certain actor groups or recommending on certain resource types, at the same time allowing grand-scale interconnection. Distributed and open architectures also allow users to design and implement their own recommending schemes and algorithms, which can interoperate with other customized systems.

Relevant uses of software agent technology in achieving tasks related to recommendation are analyzed in [113]. Namely, user modelling is needed for taking personal preferences into account in recommending. Distribution, matchmaking, and reputation and trust management can be achieved with multi-agent system technologies. Intelligent reasoning capabilities are needed for social network modelling and analysis. In the community of agent systems, there is relevant ongoing research for achieving these goals.

10.3 Recommender Systems in Travel and Tourism

The travel and tourism industry is one of the most important and dynamic sectors in business-to-consumer (b2c) e-Commerce and online transactions are rapidly increasing. According to [153], this single sector represents nearly fifty percent of the global B2C turnover. At least in developed countries, the Web is nowadays already the primary source of information for people when searching or booking suitable travel destinations and that is the trend as well in developing countries.

Products and services in the field of tourism are mainly not physical and typically exist mostly as information. For this reason, they are very adequate for electronic sale. ICT allows to easily present tourism offers with rich descriptions to enable travelers make informed choices. Therefore, the complexity of product descriptions is growing. As tourists of today are very demanding and have numerous desires and needs, tourism offers should be multi-optional and of high quality. Thus, systems that help to take these decisions on the Web become more and more significant nowadays, calling for modern means of decision-making support and recommender systems.

Tourism is an information based business, at the moment of decision-making only the description of the product is available. This characteristic of tourism products entails high information search costs and causes informational market imperfections. Consequently, the industry has comparably long information and value chains. In Figure 10.2 (extracted from [153]) the principal actors in the tourism chain are illustrated. The supply and demand sides are separated from the respective intermediary layer. Nodes indicate the relevant types of players. Links mark the relationships as well as the flow of information, showing only the most important links.

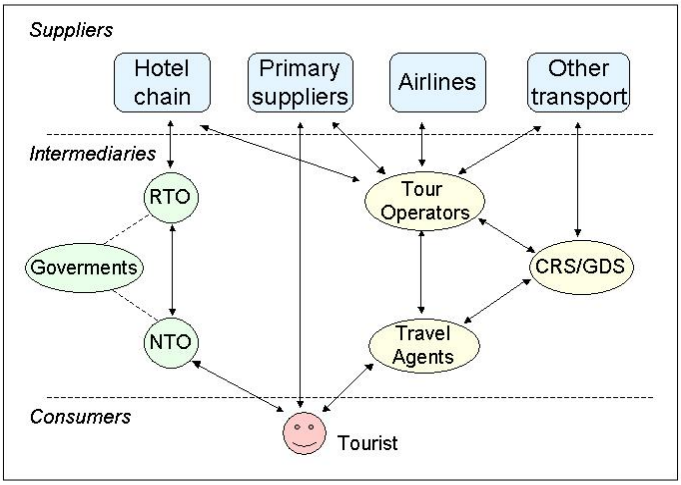

Figure 10.2: Structural view of tourism market

On the supply side, primary supplier enterprises like hotels, restaurants, etc. (which are mostly SMEs) and big companies like airlines, are placed. Tour operators can be seen as product aggregators, travel agents act as information brokers, providing final consumers with the relevant information and booking facilities. CRS/GDS (Central Reservation Systems / Global Distribution Systems), stemming from the airline reservation systems, include also other products (e.g. other means of transport). Whereas the intermediaries on the right side can be seen as the commercial connection between supply and demand, the left side is relevant for national (NTO) and regional organizations (RTO) in charge of destination planning and promotion. Normally, these entities have to act on behalf of all suppliers within a destination and are often governmental organizations. The downstream flow of Figure 10.2 consists of product information, whereas the upstream flow reports on market behaviour. Both information flows create a tourist information network relating most of the market participants.

When looking at today's e-Tourism web sites we can observe that only some of the existing systems provide services that go beyond a pure booking system's functionality. An exception are popular online travel agencies like Expedia

(www.expedia.com), that permit exploiting the potential of Web communities by letting their customers rate individual hotels or destinations. Still, in these applications the average ratings of other customers merely serve as another piece of information to chose a certain hotel or destination but there is typically no recommendation service available.

There are several reasons why established recommendation techniques cannot be directly applied to the tourism domain [57]. Collaborative filtering techniques work best when there exists a broad user community and each user has already

rated a significant number of items. As individual travel planning activities are typically much less frequent and in addition, the items themselves may have a far more complex structure, it is hard to establish reasonable user profiles. Therefore, many approaches aim at eliciting the preferences and requirements in a conversational dialog using, for example, knowledge-based approaches for generating recommendations [24, 58].

Another important facet which makes recommendation in the tourism domain more complex is the fact that a single trip arrangement may consist of several, independently configurable services. Typically, only pre-defined packages like 'flight and hotel' or 'all-inclusive' arrangements are available online. As the segment of individualized travel arrangements is constantly growing, it will be increasingly important that future systems support such packaging services. Nevertheless, only first attempts in that direction can be found in literature today (e.g. see [129]).

Consequently, the domain has always been at the forefront of information technology and still is a highly attractive research area as its potential is not yet fully exploited. The hybrid recommender approaches seems to be the best candidates in this rich and heterogeneous information domain.

10.4 Case Study: Recommender System on Argentinian Tourism

As in other developed countries, e-commerce is changing the Argentinian tourism and travel businesses. The Web is becoming an important source of information and each day an increasing number of online booking services is being added. Nowadays, the Latin American tourist behaviour has some differences with respect to other markets, e.g. the European one. While the consumer behaviour of the last group is focussed on destinations and the tourist usually plans their travels on their own, the role of travel agencies to recommend and sell tourism services and packages is key in the Latin American community. Normally a tourist turns to a travel agency to consult and choose a predefined tourism package (an itinerary including means of transport, accommodations, excursions, activities, etc). This commerce strategy is supported by providing the tourist with lower travel costs.

Thus, the focus of our case study is the recommendation of tourist packages in Argentinian destinations. The packages used as resources are provided by different tourist operators and were downloaded from the Internet.

In the previous section we have discussed the complex tourism market and its different actors. Since the principal goal of our case study is to design and develop a recommender system that allow us to experiment our g-BDI model of agent, some simplifications on the general tourism recommender problem were made. We situate our case study in the intermediate sector of the tourism market, involving travel agents and tourism operators, as can be seen in Figure 10.3.

Some characteristics for the recommender system in the case study are pointed

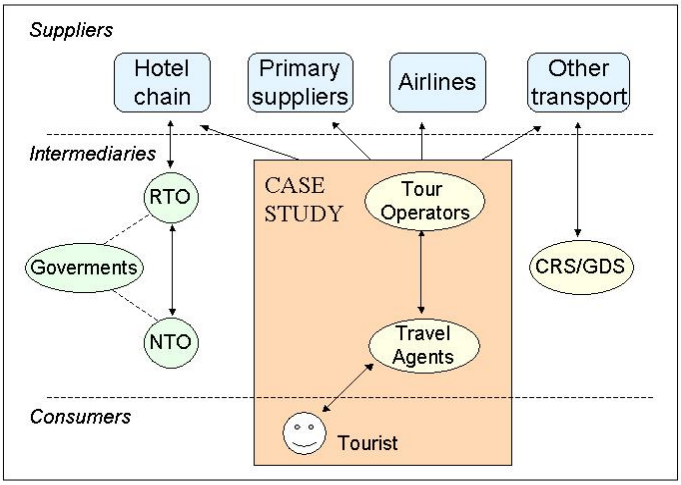

Figure 10.3: Case study in the tourism market

out in the following items:

- Resources: The resources to recommend are Argentinian tourist packages provided by different tourist operators. These packages are described by a detailed travel-destination sequence including accommodation, activities, etc.
- Recommender techniques: The recommender system will be a hybrid system that combines a knowledge base approach with a content based one. In this case the resource contents are the package descriptions and are fundamental for the recommendation. Also domain knowledge, represented by ontologies and system rules, is needed to evaluate the expected satisfaction of the tourist preferences by a tourist package.
- User profile: Each recommendation request is considered independently, and the user graded preferences and restrictions constitute the user's profile. These tourist's preferences are stored but are not used for a user profile updating process.
- Social model: a basic social model of the actors is contemplated including the reputation a travel agent has with its tourist providers, this reputation is simulated and for this case study we did not use a model of trust update.
- Descriptions: The tourist is requested to fill in a user satisfaction report about the given recommendation, stored for off-line statistical analysis and system adjustments.

In the next Chapter we design and develop this case study: a Recommender System on Argentinian Packages. This system has the purpose of select the best

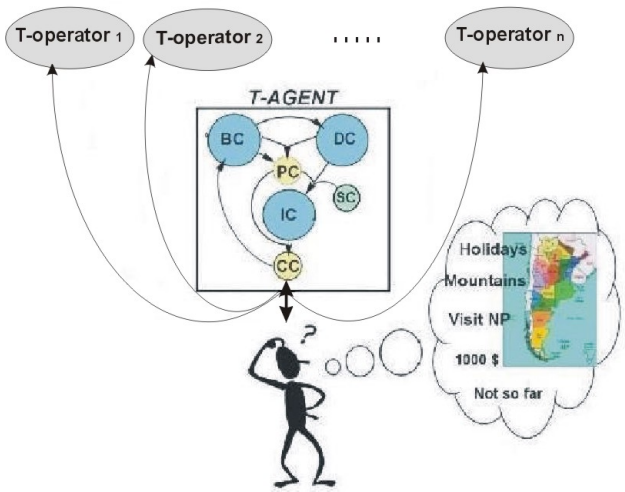

Figure 10.4: Recommender System on Argentinian Packages

tourist packages in Argentinian places according to the preferences and restrictions of a tourist, from the packages offered by diverse tourist operators. This recommender system has a multiagent architecture and one of its main agents, the Travel Assistant Agent (*T-Agent*), is modelled as a graded BDI agent (an illustration can be seen in Figure 10.4).

The ideal engineer is a composite. He is not a scientist, he is not a mathematician, he is not a sociologist... but he may use the knowledge and techniques of any or all of these disciplines in solving engineering problems.

N. W. Dougherty

Chapter 11

Methodology to Engineer g-BDI Agents and a Case Study

11.1 Introduction

Agent technology has received a great deal of attention in the last few years and, as a result, many software applications are developed using this technology. In spite of the different developed agent theories, languages, architectures and the successful agent-based applications, further work is needed for specifying (and applying) techniques to develop applications using agent technology. The role of agent-oriented methodologies is to assist in all the phases of the life cycle of an agent-based application, including its management.

Many different Agent Oriented Software Engineering (AOSE) approaches have been proposed, a survey of some of them can be seen in [13]. Each of the methodologies has different strengths and weaknesses, and diverse specialized features to support different aspects of their intended application domains. Most of the methodologies have shown that there is a conceptual level for analyzing the agent-based systems, no matter the agent theory, agent architecture or agent language they are supported by. This conceptual level should describe fundamentally the external view point of agents by the Role Models (the characteristics/tasks of each agent) and the Society Models (the relationships and interactions between the agents). We consider that it is important for a methodology to also include the agent detailed design, adopting the necessary tools to develop its architecture, as pointed in [93, 118].

A relevant architecture that provide the agent-based systems with a formal support, is the BDI architecture proposed by Rao and Georgeff [123]. The BDI paradigm provides a strong notion of agency: agents are viewed as having certain mental attitudes (Beliefs, Desires and Intentions) which represent respectively their information, motivational and deliberation states. These mental attitudes play a relevant role in the process of determining the agent actions. With the purpose of making the BDI architecture more flexible, we have proposed a general model for graded

BDI Agents described in Chapters 4 to 6.

Since there is no standard agent architecture, the design of the agents needs to be customized to each agent architecture. In this Chapter we present a methodology to engineer graded BDI (g-BDI) agent based systems.

Software Engineering for BDI Agent Based Systems

There are few works on Software Engineering for BDI Agent Based Systems. Kinny et al. in [93] proposed a methodology for agent-oriented analysis and design, focussing upon the BDI model of agents. In specifying an agent system, they have found that it is highly desirable to adopt a specialized set of models which operate at two distinct levels of abstraction. First, from the external viewpoint, the system is decomposed into agents. Second, from the internal point of view, the elements required by a particular agent architecture must be modelled for each agent.

More recently, Jo et al. in [88, 89] proposed the BDI Agent Software Development Process (BDI-ASDP) as a specialization of traditional and Object Oriented software engineering methodologies, embracing several steps enumerated below. A similar approach of software engineering process for multi-agent systems is presented by Zhang et al. in [159]. These proposals share the same approach, they take advantage of different artifacts proved to be useful in Object-Oriented Software Engineering, adapting them to their purpose. During the software analysis and design phases they define which agents integrate the system. Furthermore, some of the artifacts used in these phases support the design of the BDI architecture for each agent. Following the natural style of human thinking "goal-plan-data", these proposals first extract the desires (goals) from the requirements and create the proper plans towards them. Then, they find the beliefs. The BDI extraction process is done during the task of agent recognition and after identifying the system goals and plans. More specifically, these proposals for software modelling contain the following iterative stages:

1. They use some artifacts to specify system requirements (e.g., External Use Cases) and to extract goals (desires) from them;

2. They use Dynamic Models (e.g., Internal Use Cases, Sequence Diagrams, and Activity Diagrams) to provide a more precise description of each goal and its corresponding plan (intentions).

3. A role analysis is performed from the list of goals and their corresponding plans. The relevant roles and their interactions (role composition) are taken into account to define the set of agents in the system.

4. Finally, using Data Models (e.g., Data Flow Diagrams) they propose to obtain the environment information (beliefs) that is necessary for the goals satisfaction.

5. After a complete BDI specification has been obtained, then it is assigned to an agent.

We can remark that in both works [89, 159] there is no clear separation in the agent analysis and design between the external and the internal viewpoints. Also, the internal BDI architecture of agents is only described considering what the contents of the different attitudes (i.e., B,D,I) would be. In fact, their approaches neither present how to specify them nor show how the agent can use them to decide the current action to follow.

Besides, Sierra et. al in [141] extended the Prometheus methodology [118] emphasizing the social design of multi-agent systems. They particularly focus on a design methodology for agent societies or organizations (i.e., Electronic Institutions) where norms and rules must be abided by all the participating entities. This methodology contributes with some elements related to the social aspects of the system design.

Based on these previous works, we present a methodological framework to engineer graded BDI agents. We work up these approaches, adapting and extending them, to engineer agents with a more complex internal architecture. In this sense, we make more emphasis in the separation of the system design from the agent design phase. We also consider important to include some social aspects in the system design phase and to add some steps to support the agent detailed design.

Furthermore, in our approach we use a novel notion of intention, related to a pair desire/plan, where the desire is the goal that the agent will try to satisfy by executing the plan. The agent will consider desire/plan pairs with the best cost/benefit relation for reaching a given goal by executing a feasible plan. As a result of this analysis the agent has to decide which intention (a chosen goal) to follow by executing the best plan towards it. In an agent design, this deliberation process and the elements involved must be both formalized.

In order to design a g-BDI agent, we propose a process that starting from the external stages, where some roles and functionalities are assigned to it, moves forward the definition of the elements that compose the multi-context architecture. Next, the different stages of this process are described and a case study is used to illustrate them.

11.2 The Development Process of g-BDI Agent-Based Systems

For engineering agent-based systems, we follow the methodology presented in [93, 118] where two different design levels were defined. In one level, from the external viewpoint, the system is decomposed into agents, modelled by their roles, responsibilities and services they perform, the information they require and maintain, and their external interactions. In a second level, from the internal point of view, the elements required by a particular agent architecture must be modelled for each agent.

Then, in our approach we consider two important phases: the *System Specification and Design* (i.e., external) and the *Agent Design* (i.e., internal).

The purpose of the *System Specification and Design phase* is to establish the social structure of the system. This System phase starting from the problem statement and the social structure related to it, results in the different agent types that integrate the multi-agent system and the necessary interactions between them. This phase may be divided in two important stages: the *System Specification and Analysis*, and the *System Architecture Design*.

In the *Agent Architecture Design* stage the agent architecture for each agent type is set. Then, for its specification we need to bridge the gap from the external functionalities assigned to a particular agent, to the elements that compose the architecture, using the information extracted in the previous stages. In our case, we focus on modelling agents with the g-BDI architecture presented. Then, for its specification we need to set the logical structure and contents of the different contexts (either mental or functional) and the interactions between these units, represented by bridge rules. In the Detailed Design stage, its specification is completed in order to define concrete agents.

The flow of the overall development process for BDI agent-based systems is depicted in Figure 11.1. The principal steps, that make use of some tools, are represented in different boxes. In the *g-BDI Agent Design* stage, the boxes in dot lines represent contexts that are included in the g-BDI architecture respecting the logical model proposed and do not need any specification in this stage. The arrows illustrate the dependences between the different steps and intend to show a possible sequence between them.

In the following items we outline the principal steps of the methodology stages:

(I) System Specification and Analysis

This stage begins defining the system requirements through the Initial problem statement. This process aims at determining the system social structure (i.e., roles and actors), the system goals, and the necessary ontologic elements. The system goals are captured with the support of Use Cases and may be of different types. We take a graded view of goals (desire states), allowing to distinguish different degrees of preference between goals of the same type.

(II) System Architecture Design

In this stage, from the elements and structures found in the previous stage the necessary Agent types to structure the system are identified. From the system goals and use cases, a task structure is developed representing a plan for the goals achievement. Besides, norms and role constraints (rejected states) are extracted. We consider it important to incorporate degrees on these constraints to differentiate the level of rejection of each state.

Starting from the social structure and considering the system goals and task structures, a role composition process begins (i.e. a role has a social function and

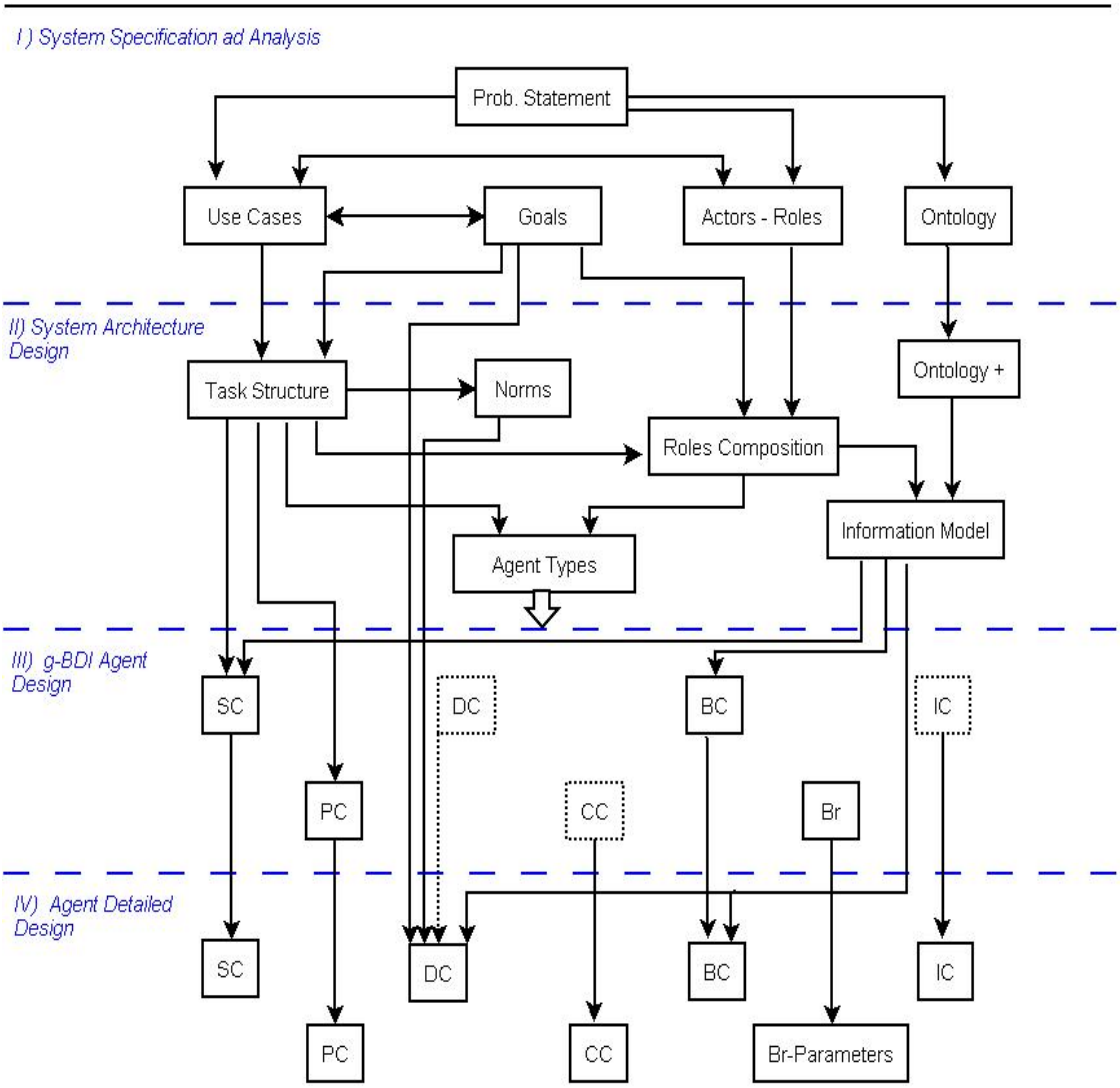

Figure 11.1: Development Process for BDI Agent-Based Systems

may have a list of goals attached to it). Furthermore, in this stage the system Information Model is built from the domain ontologies (representing the necessary information) and role interactions (determining the information flow).

The agents in the multi-agent system will be defined through the integration of the relevant role models. From an iterative role composition process, the initial list of roles may be refined to determine the candidate agents.

(III) Agent Architecture Design

Following with the design process, the architecture of each agent type must be designed. We propose to model agents using the g-BDI architecture. Considering the multi-context specification for the basic g-BDI agent model, we have proposed three mental contexts to represent the mental attitudes (i.e. Belief, Desires and Intentions) and two functional contexts (i.e., Planner and Communications). In this stage, the engineer must take the decision of whether to include another context (e.g., a social context) in the architecture and if it is the case, which is the most suitable logical schema for this context.

The g-BDI model proposes a logical framework for the different contexts (see Chapters 5 to 7), from the information extracted in the previous stages we must capture the elements to complete the logical skeleton for each agent type. Particularly, the selection of the uncertainty model for the Belief context must be done taking into account the system information model. For the Planner context, the planning algorithm must be defined from the task structure.

(IV) Agent Detailed Design

In this final design stage, the agent architecture is completed to define executable agents. This means that the contents (i.e., theories) of the mental and functional contexts, and a suitable set of bridge rules must be defined. We use the information extracted in the previous stages to fill in this agent model, this process is explained in next Section 11.4.2 and is illustrated in Figure 11.1.

Following the overall flow "goals, feasible-plans, beliefs and intention", our approach to the *g-BDI Agent Design* will first extract the agent desires from the requirements (i.e., system goals and norms). From the task structure it will analyze the possible plans towards the agent goals. Then, from the information model the beliefs involved will be captured. Finally, it will set the how to derive the agent intentions by defining a suitable Bridge rule.

Since in practice the methodology is iterative, analysts or designers may freely move between steps and phases and each successive iteration will produce additional details to finally provide a complete, yet consistent system design.

In the following subsections we describe the most important stages and steps of the software engineering process presented. To illustrate and clarify these different steps, we describe the process using a Case Study in the tourism domain (see Chapter 10).

11.3 System Analysis and Design Phase

The purpose of the *System Design phase* is to establish the social structure of the system. This phase is divided in two important stages: the *System Specification and Analysis*, and the *System Architecture Design*.

11.3.1 Stage I: System Specification and Analysis

As usual in Software Engineering, the requirement analysis is the initial part of the software development process. It will assist us to understand the purpose of the system, its social structure and how to construct it. During the system analysis, the investigation on the problem and its requirements is deepened. We focus on finding the system roles and goals. The different steps in this stage will allow in turn to extract the necessary elements for the system design that later will help in the g-BDI agent design.

Step 1: Initial Problem Statement

This is the previous and fundamental step for the System Analysis, where the problem that the system is expected to solve is described. It is a high level conceptualization of the system from the user's point of view, and describes the services that the system will provide. It is the input to capture the system goals and social structure (i.e., actors, roles, and their interactions). The initial Problem Statement is an agreement document between the user and the software developers, on a high level of description.

Case Study:

We want to design a recommender system on Argentinian tourist plans. This system will be in charge of looking for different holiday packages in Argentinian places, in order to satisfy the tourist desires. The tourist plans are described by a traveling-staying sequence. Where the different travels are described by their means of transport and kind of road, and the stays are described by their destination, accommodation, activities, etc. These tourist plans are provided by different tourist operators which in turn interact with the airlines, transport companies, hotels, and tourism services.

The customer's desires may be preferences about Argentinian zone, geographic conditions, infrastructures, activities, means of transport and accommodation. They may also have different rejections or restrictions, as for example a given maximum amount they can spend. The tourist plan the system is expected to offer, must be the best choice among the tourist packages supplied by a set of tourism operators. The system has to decide which tourist package (plan) to recommend taking into account the user's interests, the expected satisfaction of the preferences by the plan, its cost and the trust in the plan supplier.

Step 2: Actors and Roles

A natural starting point of the system analysis is to identify the system roles and actors. To establish the social structure of the system, it is necessary to capture the roles that will interact within the system, respecting the natural roles in the domain, and their relationships.

The actors are persons or entities (including other systems) external to the system, that interact with it. Some of them trigger the system behaviour to achieve a certain goal. The outcome of this step is a list of roles and actors and, if it is needed, a brief description of some of them may be included.

Case Study:

In our example we extracted two roles from the initial steps: The Provider role and the Travel Assistant role. The Provider role interacts with the different services (i.e., hotel owners, airlines, means of transports, tourism companies), builds the tourist plans (a detailed travel and stay sequence, including its cost) and sends them to a repository of tourist plans. Then, the Travel Assistant role considering the user's preferences, finds from the plan repository the best plans to recommend to the user.

The actors could be: the Tourist (user), the Airlines, the Hotel owners and other Tourist services.

Step 3: Use Cases

We can apply the *Use Cases* technique, coming from UML [17], to capture the intended behaviour of the system. A Use Case is a description of a sequence of actions, including variants that a system performs, to yield a service to an actor. Use Cases treat the system as a black box and show how the entities outside of the system interact with it. Besides capturing the system goals, the Use Case technique also describes the interaction between the system and its environment, and identifies external actors. In this step the system services are identified. The Use Case technique captures who (actor) does what (interaction) with the system, for what purpose (goal) and without dealing with the system internals. In some works, like [88], this kind of use case, is called *external use case*.

The uses cases are specified in a number of ways, from informal structure text to pseudocode. There is no standard template for documenting use cases. Typical sections may include: Use Case Name, Actors, Preconditions, Main Scenario (basic course of events), Alternative paths and Postconditions. We propose to include in the use case description, the constraints related to the behaviour that the use case describes. In an iterative process, more detailed use cases may be given. A detailed Use Case refines an specific description of a system action as for example, decomposing the global goal into sub-goals or including a plan (task structure) for achieving each goal (subgoal).

Case Study

In order to discover all the functionalities that the Recommender system should provide, we develop use cases. Next, we show one of them:

- *Name: Give a suitable recommendation*
- *Actors:* Tourist and Tourist services (e.g. hotel owners, airlines, transport companies)
- *Scenario:*
  a) The Tourist requires a personal tourist recommendation from the System.
  b) The System acquires a set of graded Tourist's preferences and rejections (restrictions) about the tourist plans.
  c) The System updates the repository of tourist packages.
  d) The System finds the best packages according to their preferences and constraints.
  e) The System gives the Tourist a ranking of the best tourist packages.
- *Preconditions:* The Systems has some tourist plans to offer. The Tourist requires from the System, a personal tourism recommendation.
- *Success/Postcondition:* The System offers the Tourist a list of ranked tourist plans.

Step 4: Goals

From the previous steps (the initial problem statement and use cases) a set of goals can be captured and structured considering possible inter-relationships between them. The extraction of system goals goes hand in hand with the identification of use cases.

The system goals may be of different types, as defined in [141]. These goals may be *individual goals* that are allocated to a role (and later to an agent type), *joint goals* that are achieved by a group of roles (eventually agents) and *social goals* setting the desired social properties of the system whose achievement the multi-agent system must ensure. In successive iterations the system goals may be refined into a subgoal structure. Each goal may be placed differently in the hierarchy (w.r.t. a global goal) and have different importance. To deal with this, we propose to use graded expressions (valued in $[0, 1]$) in order to represent the different levels of importance of goals at the same level of the hierarchy. Finally a diagram, called Goal Overview Diagram, should be used to represent the different types of goals and subgoals, including their possible relationships.

Case Study:

In our case study, we capture the following goals:

- *The social goal* "is to give a reliable service (e.g., via reliable providers, using reliable information)".

- *The overall system goal* is "to give a Tourist a suitable recommendation based on the tourist packages the system has". To achieve this system goal we divide it in some subgoals (tasks) that are shown with the task structure in the Overview Diagram of Figure 11.2.

Step 5: Ontology

During the system specification and analysis there is an identification of the data used an produced. From this analysis the ontologic model definition starts identifying the information needed in the domain.

Case Study:

For our case study, the needed ontologic elements are:
- tourism knowledge,
- knowledge about Argentinian destinations and regions, and
- tourist plans

11.3.2 Stage II: System Architecture Design

During this design stage, emphasis is put on defining software agents and on how they collaborate to fulfill the global requirements.

Step 6: Detailed Use Cases and Task structure

The purpose of this step is to get a better understanding of the role interactions based on the actions that each role (or group of roles) may execute towards the achievement of a goal. At this point some new roles may be necessary to support different internal functionalities. In this Step the different system goals are seen from an internal point of view using detailed use cases. These use cases are concerned with interactions among agents inside the system and how they use each other to get things done. The development of the detailed use cases helps to better understand the interactions and collaborations between roles, and consequently, between the candidate agents.

Case Study:

As a matter of example, we describe one detailed use case for the *Give a suitable recommendation* functionality of the *T-Agent*. This case is a detailed version of the corresponding external use case.

- *Name: Give a suitable recommendation*
- *Actors:* Tourist and Tourist services (e.g. hotel owners, airlines, transport companies)
- *Scenario:*

  a) The Tourist requires a personal tourism recommendation from the System.

  b) The Interface role acquires a set of graded user preferences and rejections (restrictions) about the tourist plans.

  c) The Travel-Assistant role asks the Provider role for tourist packages.

  d) The Repository-Maintenance role updates the package repository with the plans the Provider role sends.

  e) The Travel-Assistant role finds the best packages satisfying some of the Tourist preferences and avoiding the rejections. Each feasible plan has an associated cost. The set of feasible plans are ranked using a function that combines in a suitable way: the intensity of the desires, the Tourist expected satisfaction by a plan, the cost of the plan, and the trust in the plan provider.

  f) The Travel-Assistant role sends to the Interface role a ranking of the best plans.

  g) The Interface role shows the recommendation and the plan description to the user.

From this detailed use case we can design a task structure towards the system goal of giving a suitable recommendation. In this example, for achieving this overall system goal the first task (subgoal) is to acquire the user's preferences. Then, another subgoal is to update the tourist plan repository with the plans provided by the tourism Provider role. Next, the system must look for the best plan to satisfy the user preferences.

The global goal-task structure for this case study can be seen in Figure 11.2. This goal and its subgoals are represented by ellipses and constitute the task structure. Besides, we include in this illustration two social goals (represented by boxes) related to the system goal.

Step 7: Norms and Constraints

Norms are conditions that should be enforced, if possible, by the infrastructure of the multi-agent system. In some kind of systems (e.g., Electronic Institution) it is fundamental to control the interactions between the participants and ensure that they all adhere to agreed rules or norms. The norms are usually defined towards the end of the social design process when the structure is completely defined.

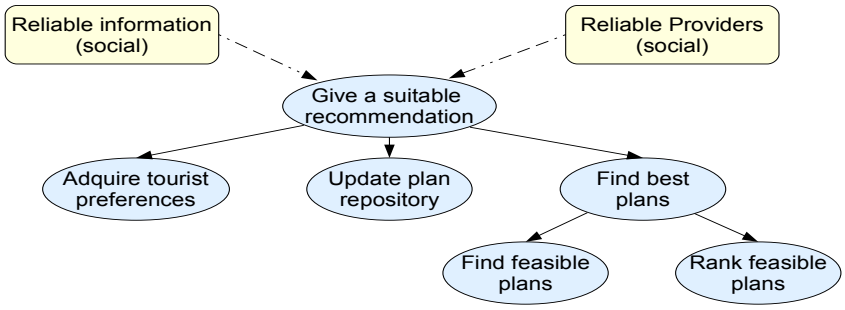

Figure 11.2: Goal-Task Overview Diagram

Constraints are also added at this stage and represent some restrictions that a role may have in a particular scenario or in relation to a goal achievement.

Case Study:
For our case study we do not extract any norm, but we consider some constraints the tourist may have in relation to the tourist package he/she is looking for. We set for this example that this restrictions may be about the days they are available to spend in their holidays, the maximum amount to spend and the kilometers to travel. Also, these restrictions may be strict or soft and may be thus treated in a different way.

Step 8: Ontology +

There is some initial ontological information specified during the System Specification phase, as the engineer identifies the information needed within a particular scenario. This initial ontology is brought into this Design step and provides the basis for more thorough refinement and development.

Case Study:

In our example, the necessary knowledge that have been identified are tourism knowledge, information about Argentinian tourist places and Argentinian tourist packages.

The tourist packages are structured following the information extracted from a set of examples, as a suitable *travel-stay* sequence. The information on Argentinian places and their characteristics are organized following a *destination ontology* structure. For each city or town, this ontology contains information about geographic and infrastructure characteristics, accommodations, activities offered, excursions and relevant issues related to the place. The needed tourism knowledge includes *similarity relations* and *rules* for deriving the belief in the preference satisfaction by the plan execution (see Chapter 12 for a detailed description).

Step 9: Role Composition

Different agent-based software engineering methodologies take advantage of a role analysis for the agents definition, as for example in [92, 157, 159, 118]. In the analysis phase we have detected some natural roles related to the social structure of the problem. This set of roles may be expanded after the extraction of the system goals. From the structure of goals and plans, a role analysis is held in order to define the agents and interactions that compose the system. Functional roles, responsibilities and goals (or services) are just descriptions of purposeful behaviours at different levels of abstraction. In our proposal, a role includes a set of goals and a role (or a set of roles) will be mapped into an agent who will be responsible for satisfying these goals.

This process of role composition and role assignment to agent types is a difficult task because is not just a distribution problem: the reasons for and against grouping particular functionalities must be carefully analyzed. This is so because there is a need to compose roles when agents carry out their responsibilities and interact/collaborate, there may be a synergy between the different roles played by the agent. More generally, we seek to have agents which have strong coherence and loose coupling. The activity of identifying roles from use cases and the use of role patterns in agent software engineering are described in [92]. The resulting combined roles are the candidate agents. The different roles, with their assigned goals or tasks and collaborators, may be gathered in a table similar to the RGC (Responsibility, Goal, Collaborator) card proposed in [159].

In agent-based modelling, interaction diagrams may be useful and some agent based software engineering methodologies adopted them (e.g. in [118, 157]). For example, some sequence diagrams are used for modelling temporal ordering of interactions, and collaboration diagrams are used to emphasize the structural organization of the agents. From this process of role identification and composition, a refined list of candidate agents is obtained.

Case Study:

Inspired in the different members of a tourism chain, in the analysis phase we have detected the following roles: the Provider role (tourist package providers), the Travel Assistant role and Services role (hotel owners, airlines, etc.). In this case study we don't deal with the Services role, we only mention it as a necessary collaborator of the Provider role.

From the task structure we capture two more roles: the Interface role, to manage the user interface and the Repository-Maintenance role (R-Maintenance), to charge and discharge the tourist packages that are sent by the Provider role. We map the goals and tasks, extracted and structured in previous steps into the different roles. For our case study, the defined roles with their assigned goals or tasks, and their collaborators (that is, other roles that interact with it) are shown in Table 11.1.

We also use an interaction model to schematize the roles interaction as can be

Roles	**Goals - Tasks**	**Collaborators-Interactions**
Interface	Acquire the Tourist's preferences Communicate best plans	Tourist Travel Assistant
Provider	Build tourist plans Communicate plans	Services R-Maintenance
R-Maintenance	Charge-Discharge tourist plans	Provider Package Repository
Travel Assistant	Find feasible plans Rank feasible plans	Interface Provider

Table 11.1: RGC card for roles

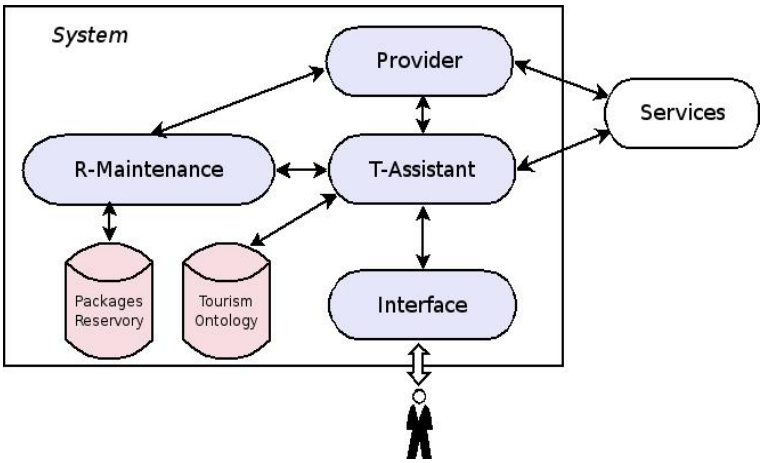

Figure 11.3: Role interaction model

shown in Figure 11.3.

Step 10: Information Model

After defining the detailed roles and their interactions, we also need to specify what information the agents filling the different roles, need. All the information that will be referenced within constraints or norms, needs also to be part of the information model. For this design process the input and output data requirements for each subgoal in a plan must be analyzed, in order to be sure that this information will be available in the needed stage. With the support of internal use cases we can extract the information needed for each action in a plan (Step 6). Besides, we may apply some artifacts (e.g. the Data Flow Diagram) to show the data flow from external entities into the agent, and how the data flows from one process to another, as well as its logical storage.

Case Study:

For our example, we extract the following necessary information:

- The different graded user preferences/rejections respect to the tourist package he/she is looking for. This information is provided by the Tourist, is captured by the Interface role and is used by the Travel Assistant role.

- The updated description of the different tourist packages. This information is provided by the Provider role and we create a Tourist Packages Repository to gather this information. This repository is updated by the R-Maintenance role and is used by the Travel Assistant role.

- General knowledge about tourism and information about Argentinian regions and destinations. The general tourism knowledge is represented by special relations and rules. The Argentinian information is structured in a suitable tourism ontology (organized by destinations), and it is maintained and used by the Travel Assistant role.

The information storages interacting with the different system roles are illustrated in Figure 11.3.

Final Step: Agent Types Definition

From the iterative process of role identification and composition, supported by some interaction and information models, the list of the candidate agents is finally defined.

Case Study:

For this prototype Recommender System, we define only two agent types: the Provider Agent and the Travel Assistant Agent. We assign the Interface role, the Repository Maintenance role and the Travel Assistant role to the Travel Assistant Agent (*T-Agent*). As it is natural in the Tourism Chain, different Tourist Operators may collaborate in the Provider role. To represent these different sources of tourist packages, we use different agents (Provider Agents).

Then, the agents composing our multi-agent recommender system are: the *T-Agent* and a finite set of Provider Agents (Provider-i, i=1,...,n). An agent interaction diagram for this multi-agent system is illustrated in Figure 11.4.

11.4 Agent Design Phase

Once the different agent types in the system are defined, we have to deal with their internal design. Namely, we must decide what kind of agent architecture is the most appropriate in each case according to its characteristics and its role assignment. In this Section we focus on the methodological process to develop agents using the g-BDI model. This design process is done in two stages: the first stage is concerned with completing the logical skeleton of the agent architecture, on the second one,

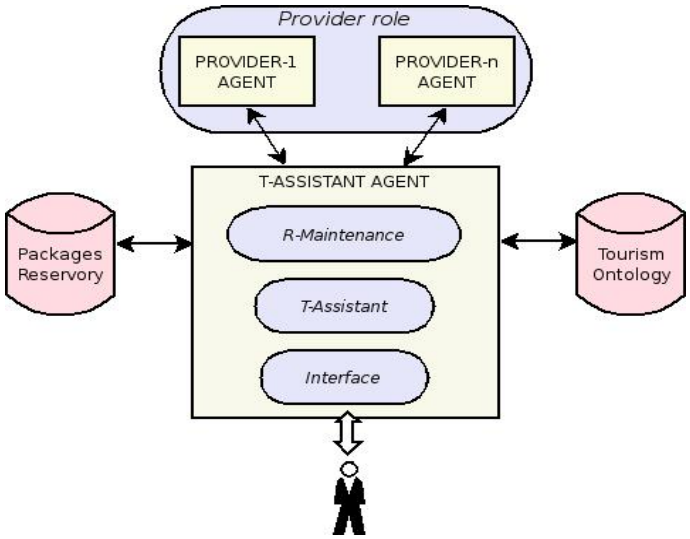

Figure 11.4: The multi-agent Recommender System

called *Detailed Design stage*, the contents (theories) of each context are defined. To illustrate how this methodology works, we show the detailed design of the Travel Assistant Agent (*T-Agent*).

11.4.1 Stage III: A graded BDI Agent Design

In this Stage we show the methodology used to develop agents following the g-BDI model. Taking advantage of its multi-context approach, this amounts to specify the different contexts, either mental, functional and the Bridge rules.

In this stage we must go from functionalities assigned to an agent type to a multi-context model, conforming an agent capable to reach each desired goal (goals). For the modelling of the different contexts, we will use the information acquired in the system analysis and design stages. Next, some aspects related to the different contexts are analyzed to complete the multi-context schema for an agent type.

1- Social Context

It must be decided whether or not to include a social context in the agent architecture to represent the social aspects of agency. A key issue related to the social aspects is the modelling of the agent trust in other agents. In an agent community different kinds of trust may be needed and should be modelled. The necessity of a social context in an agent model depends on the roles and tasks assigned to the agent, and on the information model. An overview of an appropriate logical framework to represent and reason about trust or reputation in a social context, has been proposed in the socialization of the g-BDI model (see Chapter 8).

2- Desire Context

Besides a general logical framework to represent desires in the g-BDI agent model, different logic schemas have been proposed (see Chapter 6) to represent the agent positive and negative desires. According to the agent type (roles and tasks assignment) we must decide which constraints we want to set between positive and negative desires (i.e. preferences and restrictions) respect to a formula and its negation. Thus, the most suitable logic schema may be chosen.

3- Belief Context

This context represents the agent knowledge about the world. Depending on the agent environment and on how the agent acquires information from it, the knowledge may be of different kinds: uncertain, imprecise, incomplete, etc. In the case of having uncertain information, we propose to use in the g-BDI model a fuzzy modal approach to deal with graded information and to consider a suitable uncertainty model on top of this logical framework (see Chapter 5). To complete the logical schema for this context, the selection of the uncertainty model must be done. We must decide whether to use probability, possibility or necessity measures, among other options. For this purpose, the system information model is used and two factors that may be taken into account are:

- *The source of the uncertainty information:* if the uncertainty information comes from data bases containing and important amount of data that may be statistically processed, the use of probability measures is recommended. If we only have order relations expressing which data is more certain than others, then a possibility measure is adequate.

- *The different intuitions about the expected results on operators* (e.g., conjunctions and disjunctions) also helps in the uncertainty model selection. For example, if the conjunction is expected to be the minimum of the individual values of uncertainty, we may use a necessity function.

4- Planner Context

From the task structure related to the agent goals, it can be determined how the agent will make some plans (feasible plans) to fulfill the positive desires and to avoid the negative ones. Precomputed plans or a particular planning algorithm may be used to support the Planner strategy to find feasible plans for the agent.

5- Bridge Rules

The interactions between the different contexts are represented by Bridge rules (BRs). As each unit uses a proper logic, these rules allow to embed results from a theory into another [68]. Besides, diverse agent's personalities may be modelled using a suitable set of BRs to represent, for example, different realisms (see Section 7.2).

Case Study:
For the design of the *T-Agent* using the g-BDI model, we decided the following:

1. To incorporate a Social Context to represent the trust in the Provider Agents in relation to the different tourist packages they offer.

2. To represent desires in the Desire Context, we chose the DC_3 Schema that models a strong consistency condition between desires and rejections. It represents the following restriction: if a state of the world is rejected to some extend, it cannot be positively desired at all and conversely, if a goal is somewhat desired it cannot be rejected. We found this schema is suitable to our case study.

3. To represent uncertainty in the Belief Context by using the probabilistic model, as there is quantifiable data in the tourism domain.

4. To make the Planner Context look for feasible plans in a Repository of Tourist Packages (pre-computed plans).

5. To define for the *T-Agent* a set of BRs that are needed to export formulae from one context to others. The *T-Agent* personality is represented by a particular BR, where the intention degree is computed. Depending on the tourist preferences with respect to his priority criterion in the package selection (i.e. minimum cost, preference satisfaction or trust) the BR representing the Tourist personality, will use different aggregation functions for the variables involved to compute the intention degree.

11.4.2 Stage IV: Agent Detailed Design

In this stage, we must complete the agent design process defining what the contents of each unit will be. We extract information from previous steps in the overall design process to fulfill this internal stage as it is shown in Figure 11.1. The flow of this detailed multi-context design process is illustrated in Figure 11.5. Next, we depict this process by using the case study.

Desire Context (DC):

The agent positive desires are extracted from the goals assigned to the agent. We take as desires only the proactive goals, not those subgoals that can be derived from a planning process. The positive desires are extracted from the goal and task structure (Step 4 and Step 6) related to the different roles (Step 9), that are later on assigned to the different agent types. We also consider important for the agent to include a set of negative desires, representing its rejection states. The negative desires are expressed in the use cases as part of its description and later are separated

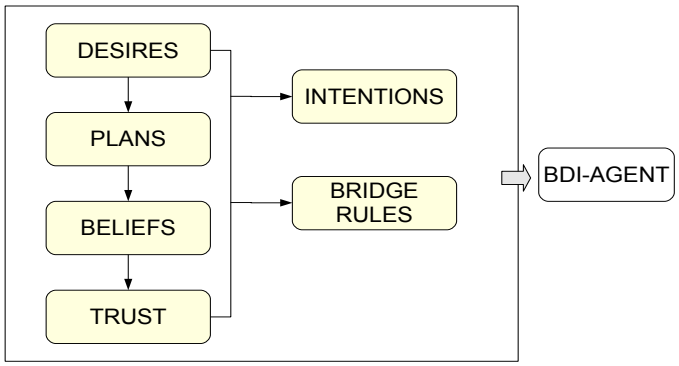

Figure 11.5: Detailed modelling process flow

as constraints (Step 7). As we mentioned in the system analysis, the set of desires (positive and negative) may have different levels of importance. Usually, these levels may be captured in a numerical scale (or in an order relation) and then, may be translated into degrees in $[0, 1]$, using an order preserving function. To represent and reason with these positive and negative graded desires, we can use the general logic framework or one of the different schemas presented in Chapter 6.

Case Study:

The *T-Agent* goal is to satisfy the tourist preferences by recommending the most suitable tourist package. As a personal agent, the *T-Agent* takes as subgoals to satisfy the different tourist preferences. Thus, the contents of the Desire context will be obtained at runtime.

The tourist desires will be expressed by a theory in the DC containing quantitative expressions about positive and negative preferences. These formulae express what the tourist desires or rejects in different degrees for his holidays. From this set of desires the DC generates all the possible conjunctions of the positive desires. For this case, we take as the negative desire the conjunction of all the rejections. An example of how the *T-Agent* desires are built from a tourist preferences can be seen in Chapter 6 (Example 1).

Planner Context (PC):

Some authors as [89, 159] directly relate the plans towards the different goals to the notion of intentions. We consider a more complex notion of the agent intention that involves the pair *feasible plan-desire*. That is, the agent will intend to reach a desire by executing the feasible plan that best satisfies a cost/benefit relation. Indeed, the existence of a plan constitutes a necessary condition to determine the agent intention. The plans towards the different goals are outlined in the initial use cases, are refined successively in more detailed use cases and are represented in the Task Structure (Step 6). Using the Task Structure, we find the set of actions that the agent may follow to reach a goal or set of goals (positive desires), satisfying the

norms and avoiding the rejections (negative desires). These plans are composed by the elementary actions that the agent can perform. These actions are part of the agent beliefs. Starting from these actions a planning process may be held, using a planning algorithm or pre-computed plans (determined in the previous stage) to find the *feasible plans*. Namely, the current state of the world must satisfy its pre-conditions, the plan must make true the positive desire the plan is built for, and cannot have any negative desire as post-condition. These feasible plans are computed within this unit taking into account beliefs and desires injected from other units (i.e., BC and DC).

Case Study:

The Planner context in the *T-Agent* is in charge of looking for tourist packages that are expected to satisfy the tourist preferences (feasible plans) in a repository of the tourist packages offered by different Provider agents. After analyzing the information of various tourist packages, we structure them including the Tourism Operator provider, the cost and itinerary description, as follows:

$$package ::= (ID, Operator, Cost, [travel_1, stay_1, ..., travel_n, stay_n, travel_{n+1}])$$

where *travel* is a description of the travel characteristics (e.g. type of transportation, travel length, etc.) and *stay* includes destination, number of days, type of accommodation and activities. Each *travel* and *stay* is considered as an atomic sub-plan amenable to satisfy desires. Packages are modelled as composed plans, alternating travel and stay sub-plans.

Belief Context (BC):

Beliefs represent the (uncertain) knowledge about the agent state and the changing environment. This knowledge is used to derive conclusions about whether a plan may fulfill some agent goals (desires). The agent beliefs are the necessary information in relation to its goals assignment, and are extracted from the Information Model (Step 10).

Case Study:

From the Information Model we extract the BC theory for the *T-Agent*, it contains at least:

- General knowledge about tourism and Argentinian regions and destinations, including the characteristic of each region and activities allowed in each place. The knowledge about destinations is structured as follows:

Destination	::=	*(Name, Coordinates, Zone, [NaturalResource], [ArtificialResource], [Activity])*
Coordinates	::=	*(X, Y)*
NaturalResource	::=	*Resource*
ArtificialResource	::=	*Resource*
Resource	::=	*(KindOfResource, Name)*

- Information about the tourist packages that the different operators provide (their structure was presented previously). This information is placed in a suitable repository and is made accessible by a BR to the PC in order to find feasible plans.

- Beliefs about how possible desires D (e.g. going to a mountain place or making rafting) are satisfied after executing different tourist plans. Following the model presented, the truth-value of $B([\alpha]D)$ is the probability of having D after executing plan α. If a package α is composed by a number of subplans α_i, $B([\alpha]D)$ will result from the probabilities r_i of having D after the execution of the sub-plan α_i, by using an appropriate aggregation operator.

Besides, different kinds of knowledge are represented in this context. On the one hand, we use bi-valued formulae to represent some tourist knowledge as, for example, the distance between two destinations. On the other hand, many-valued modal formulae are used to represent uncertain knowledge. For instance, the formula *(B[Atuel7]rafting, 0.9)* expresses that the probability of satisfying the goal of making rafting, as a consequence of following the tourist plan *Atuel7*, is greater than 0.9.

Social Context (SC)

The aim of including a SC in an agent architecture is to model the social aspects of agency. Once the trust model has been defined (to filtering incoming information or to evaluate the risk on delegation of actions, etc), this context must be filled with formulae expressing the initial trust in other agents.

Case Study:

We consider the trust in the tourist package suppliers that interact with the *T-Agent*, in order to evaluate the risk in the recommended tourist plans.

The theory for the SC in the *T-Agent* has formulae like $(T_j[\alpha]\varphi, t)$ expressing that the trust of the *T-Agent* towards a $Provider_j$ of the plan α directed to satisfy a goal φ, has degree greater than t. For this application, we consider that the trust depends only on the kind of tourist package that a Provider offers. Hence, we have proposed a plan classification based on a tourism ontology. For instance, we consider the region of the country as a classification element, since there are tour-operators that are good for plans in a particular region, but not in others. We consider that

is important for the *T-Agent* to evaluate the trust in the different plan providers, to decide which package to recommend. Thus, we introduce the trust degree as another variable that must be weighted in the computation of the intention degree, next described.

Intention Context (IC)

To complete the agent design we need to specify how the agent intentions are decided. In the g-BDI model the intention of an agent is a pair desire/plan (the desire she decides to follow by executing the plan) that is determined following a deliberation process. We consider that this attitude depends on different factors, we select the following relevant ones:

- the degree of the desire intended to be satisfied,
- the expected satisfaction degree of the desire through the plan execution,
- the cost of the plan,
- if some collaboration of other agents is involved in the plan execution, the trust in those agents must also be considered.

Different kinds of agents may be defined according to the way these elements are combined and weighted. How to do this in a suitable way is a difficult and domain dependent problem.

The intention degree is computed using a suitable bridge rule, as the formulae representing the different factors are coming from various contexts.

Case Study:

A theory for IC in the *T-Agent* represents those desires the user can intend to satisfy by different feasible plans. This theory is initially empty and will receive from a suitable bridge rule (see 11.1 in next subsection Bridge Rules) intention formulae like $(I_\alpha\varphi, i)$ for all the desires φ and for all the feasible packages α that the Planner finds to achieve them. Using this set of graded intentions, the *T-Agent* derives the final intention and the most recommended tourist plan.

Then, if the Intention with the maximum degree is obtained by the execution of the plan α_b, this package will be recommended to the user.

Communication Context (CC)

This context makes it possible to encapsulate the agent internal structure by having a unique and well-defined interface with the environment. The necessary communication among the agent and other interacting agents, may be extracted from the role interaction model and the information model (Steps 9 and 10). As in the PC context we propose to use classical first order logic (see Chapter 7). The theory of this context must be in charge of the agent communication with the other agents and entities in the multi-agent society where the agent lives.

Case Study:

The theory of this CC context takes care of the sending and receiving messages to and from the Provider agents and manages the user's interface (detailed in next Chapter 12.

Bridge Rules (BRs)

The design of the BRs is done simultaneously to the process of determining the agent intentions. For example, the set of positive and negative desires must be passed from the DC to the PC which is in charge of finding the feasible plans to satisfy these desires. Also, some beliefs are also needed by the PC context to find these plans, like the elementary actions that the agent can execute, or the beliefs in the satisfaction of the different desires after executing a plan. Furthermore, there is a fundamental BR in charge of computing the intention degrees. By defining this BR in different ways, diverse agent's personalities may be modelled. The necessary BRs for an agent specification will depend on the agent type and its role assignment, a set of basic BRs was presented in Chapter 7.

Case Study:

For the *T-Agent* we have defined a set of bridge rules, modelling the inter-context inferences. Particularly, there is a bridge rule that infers the intention degree of $I_\alpha\varphi$ for each feasible plan α that allows to achieve the goal φ.

This value is deduced by using a suitable function that combines different factors. From the degree d of the desire on φ ($D^+\varphi$) and the degree r of belief on satisfying φ by executing α ($B[\alpha]\varphi$), the *T-Agent* computes the expected satisfaction degree e after executing the plan α towards D: $E(D,\alpha)$ (see for details next Chapter 12). This expected satisfaction degree, together with the cost c of the plan α and the trust t in the tourist supplier o of the plan,[1] derive the graded intentions by using the following rule:

$$\frac{PC : fplan(D,\alpha,o,P,A,c), IC : (E(D,\alpha),e), SC : (T_o[\alpha]\varphi,t)}{IC : (I_\alpha D, f(e,c,t))} \tag{11.1}$$

Different functions f allow to model different agent behaviours. For instance, the function might be defined as a weighted average, where the different weights are set according to the user's priority interests (e.g. minimum cost, preference satisfaction or trust).

Specifying a graded BDI Agent

Collecting the specification of the different contexts and the bridge rules, the

[1]We have added an argument (o) to the predicate *fplan* used in the PC (respect to the defined in Chapter 7) to represent the plan provider.

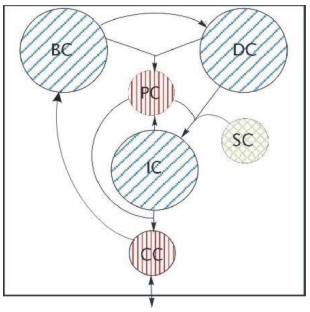

Figure 11.6: Multi-context model of the graded BDI *T-Agent*

g-BDI specification for the *T-Agent* is completed. Figure 11.6 illustrates this architecture for the *T-Agent*.

11.5 Conclusions

Agent-based computing has increased in the last years and thus the software engineering methodologies to develop these systems has become an important need. Even though there are valuable approaches in this field, few of them emphasize in the internal design of agents. In this Chapter we have presented some contributions in this direction, proposing a software engineering process to develop graded BDI agents in a multiagent scenario. The methodology presented has been built adapting and extending previous approaches [88, 118, 159] in order to engineer agents with a more complex internal architecture. Our work was also inspired by the design process described in [141] where the social aspects of design are considered, and the system design phase is clearly separated from the agent design phase.

The system design phase has the purpose of determining the agent types composing the system and it follows a similar schema than other methodologies [87, 92, 118, 159]. In the Agent Design phase we focus on modelling g-BDI agents. We extract the necessary elements from the system design phase to design the different types of agents using the proposed architecture. This process is done in two stages. The first one deals with the logical skeleton of the multi-context specification of the g-BDI model. The second one, following a flow "goals-feasible plans-beliefs-intentions" complete the agent design, filling the contents (theories) of the different contexts. In this sense, we have modified the flow "goals-plans(intentions)-beliefs" used in approaches like [88, 159] presenting a new process to obtain the agents intentions.

Furthermore, the proposed process to develop g-BDI agents contributes to bridge the gap from the external functionalities assigned to a particular agent, to the elements that composed each architecture. Particularly, we have presented the methodology applied to the design of graded BDI agents, extending the BDI model with the capabilities of dealing with the environment uncertainty and with graded mental attitudes.

In theory there is no difference between theory and practice. In practice, there is.

J. L. van de Snepscheut/Y. Berra

Chapter 12

Recommender System Implementation

In this Chapter the principal characteristics of the recommender system implementation are described. The system design was presented in Chapter 11. Especially we focus on the *T-Agent* implementation, as this agent is modelled using our g-BDI architecture. In the following sections the most relevant implementational challenges are presented, going from theory to practice.

12.1 Introduction

The implementation of a prototype of an Argentinian Recommender System is described in this Chapter. The system goal is to recommend the best tourist packages on Argentinian destination according to the user's preferences and restrictions. The packages are provided by different tourist operators. The recommender system has been designed previously (see Chapter 11 for details) using a multi-agent architecture composed by a Travel Assistant Agent (*T-Agent*) and two Provider Agents (*P-Agents*) In our implementation the Provider agents simulate different tourism Operators that supply the *T-Agent* of tourist packages. As usual in real world operators, these agents may manage the package information in different ways and using diverse formats.

The purpose of this prototype implementation is to show that the g-BDI agent model is useful to develop concrete agents on real domain. Thus, we focus on the most relevant implementational aspects of the *T-Agent* designed as a g-BDI agent. The different components in the multi-context architecture of the *T-Agent* (i.e., context and bridge rules) with their logical structure are then implemented. Particularly we show how the *T-Agent*, takes advantage of the tourist's preferences and the domain knowledge (about tourism and Argentinian places) to give the user a good recommendation.

12.2 Multiagent development

In this simplified version of Recommender System we define two agent's types: the Provider agent and the Travel Assistant Agent. In this simplified version of Recommender System, we define two agent's types: the Provider agent and the Travel Assistant Agent. We assign the interface role, the repository maintenance role and the travel assistant role to the Travel Assistant Agent (T-Agent). As it is natural in the Tourism Chain, different Tourist Operators may collaborate in the Provider role. To represent these different sources of tourist packages, we use two different provider agents (P-Agents). The internal architecture of the Provider agents is not considered in our implementation and for our purposes they are considered only tourist packages suppliers.

The multi-agent architecture of the prototype version of the tourism recommender system, composed by the T-Agent and two Provider Agents, together with the main source of information they interact with (the destination ontology and the package repository) is illustrated in Figure 11.4 of Chapter 12. This multiagent system is easily scalable to include other providers.

The implementation of the Recommender system was developed using SWI-Prolog [1]. We decided to use prolog because is a suitable language to deal with logical deduction, which is the nature of the inference processes in our agent model. Also, SWI-Prolog is a multi-threaded version of prolog allowing an independent execution of the different contexts (i.e. in different threads). Furthermore, this prolog version is open source, it is well documented and includes a graphic interface tool in native language. A previous implementation of multi-context agents using this software [69] was a starting point for our development. Furthermore, this prolog version is open source, it is well documented and includes a graphic interface tool in native-language.

In our multiagent recommender system the two Tour Operator agents (*P-Agents*) implemented runs in a different thread, so in this way being independent from each other and from the *T-Agent*. When the *T-Agent* requests for information, the *P-Agents* send to *T-Agent* all the current packages they can offer. The communication between agents is by message exchange.

In the real world, each tourist operator may structure the tourist packages in a different way and using its own terminology. To experiment with heterogeneous providers, we use different field names in the plan structure used in each *P-Agent*. Then, these structures are translated into the format the *T-Agent* uses. Thus, a wrapper functionality is needed and it is carried out by the Communication context of the *T-Agent*. In a more complete multiagent recommender architecture a wrapper agent may be included.

[1] http://www.swi-prolog.org

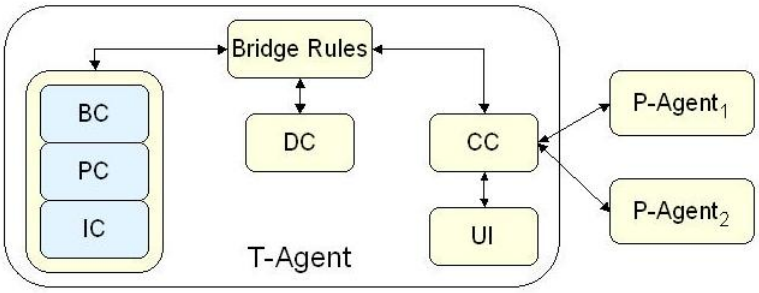

Figure 12.1: Multithread system scheme

12.3 *T-Agent* Implementation

The main role of the *T-Agent* is to provide tourists with recommendations about Argentinian packages. This agent may be suitable modelled as an intentional agent and particularly, by a *g-BDI agent model*. This agent model is specified by a multi-context architecture having mental and functional contexts (i.e, *BC, DC, IC, PC and CC*) and a set of bridge rules (*BR*s). Thus, the implementation of these interconnected components is needed. Even though, in the *T-Agent* design we have included a social context (SC) with the aim of modelling the trust or reputation in the different providers agents, in this prototype we have left out the implementation of this context.

Each context has its own inference rules and theories, and they should not interfere. Using a thread for each context allows the desired separation but could considerably slow down the execution. The solution adopted for our implementation was to place only some of these components in different threads. That is the case for the Communication context (CC), the Desire context (DC) and some bridge rules. However, since the Belief (BC), Planner (PC) and Intention (IC) contexts interchange quite a lot of information, for efficiency reasons they run in the same thread. The multithread scheme for the *T-Agent* in the multiagent system is illustrated in Figure 12.1, where the yellow boxes represent different threads and the arrows their interactions.

For this multithreaded implementation, following [69], the policy adopted is to have asynchronous threads and asynchronous communication. It means that the messages are sent and received at any time, but they are processed only when the unit is inactive (i.e. when it has finished the internal deductions). Each unit has a message queue that retains the messages until they have been processed. A communication meta-interpreter is devoted to synchronize the ongoing inference process and the arrival of new incoming messages. In our prototype, the exchange of most part of the messages is made during the initial phase. In this phase the *T-Agent* asks the *P-Agents* for the current tourist packages. To answer this request, the *P-Agents* send back a number of messages, each one containing an offered package.

The software tool successfully supports this intensive message exchange.

The Communication context (CC) in the *T-Agent* is in charge of receiving these messages, it translates them into a suitable format and it immediately sends them to the Belief context (BC). In this way the agent's knowledge is increased with the package information. In the next sections we described how the main multi-context components of the *T-Agent* are implemented in order to obtain the desired behavior. We begin with the Communication context that provides the agent with a unique and well-defined interface with the environment.

12.3.1 Communication Context

The Communication context (CC) constitutes the *T-Agent* interface and makes it possible to encapsulate the agent's internal structure. This context takes care of the sending and receiving of messages to and from other agents in the multiagent society where our graded BDI agents lives. The CC in the *T-Agent* is in charge of interacting with the tourist operators (*P-Agents*) and with the tourist user that is looking for a recommendation.

Interaction with the *P-Agents*

Before beginning its recommendation task, the *T-Agent* updates its information about current packages (carrying out its repository maintenance role). This is achieved by the CC through the following steps:

- **Requiring the packages offered:** The CC sends a message to each P-Agent asking them for the current touristic packages they offer.
- **Receiving packages and formatting them:** As the information coming from each *P-Agent* has different format the CC behaves as a wrapper, translating the incoming packages into the T-Agent format. This functionality for one of the Provider agents (agentTradingTour) is coded as follows:

```
run :-
  repeat,
    thread_get_message(X),
    parse(X),
  fail.
parse(tell(agentTradingTour, agentT,
  paq(codigo(Id), precio(Costo), Recorrido)))
  :--
  thread_send_message(brUnit, paq(id(Id),
   empresa(agentTradingTour), costo(Costo), Recorrido)).
```

- **Sending packages:** Once the packages are put under the correct format, they are sent to the Planner context. The recommendation will be based on the information about packages and on domain knowledge.

User interface

The user interface is in charge of explicitly acquiring the tourist's profile, providing him with the resulting recommendation and receiving the user's feedback. In a first approach this interface was developed using the native language. Later on, to facilitate the access to the recommender system it has been implemented as a Web service[2]. This interface process goes through the following sequential steps:

- **Acquiring user's preferences:** User's preferences are explicitly acquired asking him to fill in a form. The tourist can specify his preferences (positive desires) and restrictions (negative desires), assigning them a natural number from 1 (minimum) to 10 (maximum) to represent the level of preference or rejection in the selected item. Furthermore, he can choose different parameters: the flexibility of restrictions (by specifying them as flexible or hard), the expected frequency of the selected activity (high or low) and the priority criterion to rank-order the recommended packages (preference satisfaction or minimum cost). An example of a tourist's preferences specification using this interface is shown in

Figure 12.2: User interface: tourist's preferences.

Once the user finishes his specification, the CC sends all the acquired information to the Desire context (DC).

[2]http://musje.iiia.csic/eric/

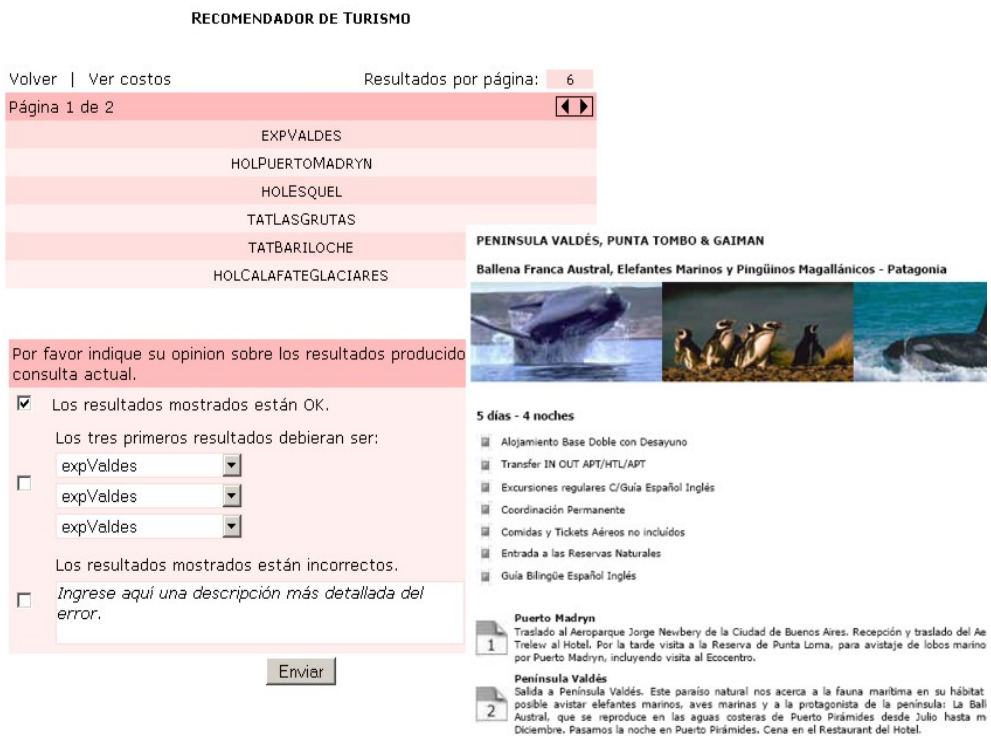

Figure 12.3: User interface: package recommendation, a package description and user feedback.

- **Showing the resulting recommendation:** As result of the *T-Agent* deliberation process, the CC receives from the Intention context (IC) a ranking of feasible packages that satisfies some of the tourist preferences. The ranking is ordered also taking into account the priority criterion he has selected (e.g. preference satisfaction). The first nine packages of this ranking are shown to the tourist and the user can visualize the information about them by opening suitable files (e.g *pdf* files). In Figure 12.3 can be seen the results for the query shown in Figure 12.2.

- **Receiving user's feedback:** After analyzing the ranking of the recommended packages the user can express through the interface his opinion about the recommendation. For this task, the options considered are the following:

  - *Correct:* when the user is satisfied with the ranking obtained.

  - *Different order:* when the recommended packages are fine for the user, but they are ranked in a different order than the user's own order. In such a case, the user is able to introduce the three best packages in the right order.

  - *Incorrect:* when the user is not satisfied with the given recommendation. Then, the interface enables him to introduce a (textual) comment about his

opinion.

All the information resulting from the previous steps (i.e., the tourist's preferences, the recommendation given and the user's feedback) is stored to evaluate the system performance.

12.3.2 Desire Context

As the *T-Agent* is a kind of *personal agent*, its overall desire is to maximize the satisfaction of tourist's preferences. Then, in this context the different tourist's graded preferences and restrictions are respectively represented as positive and negative desires.

On the one hand, the negative desires are used as strong constraints, namely, the T-Agent will discard those packages not satisfying them. On the other hand, from the elementary positive desires all their conjunctions are built as combined desires. The *T-Agent* will use all these desires as pro-active elements, looking for different packages that will allow tourists to satisfy any of them. Then, the theory in this context is constituted by positive and negative desires (represented by *desU* formulae).

The user's preferences are acquired in the CC by the user interface and are introduced in a list to the DC. In the following items we describe how the positive desires are built (negative desires are treated in a similar way):

1. **Elementary desires:** The DC takes each desire from the list received from the CC, normalizes its degree (i.e. mapping it from $\{1, ..., 10\}$ into $(0, 1]$) and adds it to the context formulae. The structure of these formulae is:

   $$desU(y(Desire, Value), NormalizedDegree)$$

   The relation $y(Desire, Value)$ represents a positive desire where the first argument is the class of desire (e.g. transport "transporte") and the second is the value the tourist has chosen (e.g. plane "avion"), followed by the normalized degree. For instance, the formal expression in the DC of the elementary desires corresponding to the tourist's preferences specified in Figure 4 are:

   ```
   desU(y(zona, patagonia), 0.9)
   desU(y(transporte, avion), 0.7)
   desU(y(comodidad, apart), 0.6)
   ```

2. **Similar Desires:** In the special case of some types of desires (e.g. those about accommodation, transportation, natural resources), we consider that a tourist can also be satisfied (to some lower degree) with a package that offers similar facilities to the ones originally specified. Also, in the particular case

of accommodation, we assume tourists will be also satisfied if they receive a better accommodation than the selected one.

Therefore, the Belief context contains instances of the domain dependent relations "*to be similar to*" and "*to be better than*" which are used to expand the set of possible values that would satisfy the user's preferences.

Indeed, these instances are used to generate new desires into the DC by means of rules like "If the T-Agent has a positive desire X at least to a degree d and he believes that X is similar to Y at least to a degree s, then he also desires Y at least to a degree $d' = d \cdot s$" and "If the T-Agent has a preference about an accommodation X at least to the degree d and Z is an accommodation *better than* ("mejorQue") X, then he also desires Z at least to degree d". These rules are formalized using suitable bridge rules.

```
desU(y(Deseo, X), d),
belU(similar(X, Y,s))
--: desU(y(Deseo, Y), d.s)

desU(y(comodidad, X), d),
belU(mejorQue (Z, X))
--: desU(y(Deseo, Z),d)
```

This rules are used for the expected satisfaction computation (see next subsection 12.6.1).

3. **Combined Desires:** After the elementary desires are added to the context, all possible conjunctions are built. The conjunctions are attached a degree greater or equal than the maximum of elementary degrees, and hence is computed in accordance with the guaranteed possibility model (see [11]). Namely, if the DC contains formulae like $desU(y(D_1), G_1)$ and $desU(y(D_2), G_2)$, a combined desire $desU(yLst([D_1, D_2]), G)$ is also added. In our prototype G is computed by the following function:

```
calcularGraduacion(G1, G2, G) :-
  G is G1 + ((1 - G1) * G2)
```

As way of example, we show the code of one of the conjunctive combinations built from the elementary desires given above:

```
desU(yLst([(zona, patagonia),
          (transporte, avion)]), 0.97)
desU(yLst([(zona, patagonia),
(comodidad, apart)]), 0.96)
desU(yLst([(transporte, avion),
(comodidad, apart)]), 0.88)
desU(yLst([(zona, patagonia), (transporte, avion),
          (comodidad, apart)]), 0.988)
```

Both, the positive and negative desires are passed by means of a bridge rule to the Planner context where the feasible packages that satisfy the tourist's preferences are selected.

12.4 Belief Context

In the Belief context the *T-Agent* represents all the necessary domain knowledge about tourism and in particular about tourism in Argentina: tourist packages, information about destinations and some special domain-dependent relations. Also, in this context the belief degrees of achieving the different desires by executing alternative plans, are computed. We describe next the representation of these kinds of information.

12.4.1 Tourist packages

One of the most significant data structures in our system is the package structure. After analyzing nearly forty Argentinian packages selected from the Internet, a general structure which is capable of representing the information available in most of them has been adopted. Each package is represented as a list containing an identifier, a tour provider, the cost and a travel-stay sequence as it can be seen in the following structure:

Package	::=	(Id, Provider, Cost, Trip)
Trip	::=	[(Travel, Stay)]
Travel	::=	(Transport, Road)
Stay	::=	(Destination, Days, Accommodation, [Activity])
Activity	::=	activity(Sport, Hours) | excursion(Resource, Hours, Name)

For example, the prolog representation of the package named *holCalafatePatagonia* is presented below:

```
paq(id(holCalafatePatagonia), costo(1900),
  [(viaje(avion, aire), estadia(calafate, dias(3), comodidad(apart),
    actividades([
      [act(cityTour), horas(4)],
      [exc(parqueNacional), horas(8), peritoMoreno]]))),
  (viaje(avion, aire), estadia(ushuaia, dias(4), comodidad(hotel3),
    actividades([
      [act(cityTour), horas(1.5)],
      [exc(museo), horas(1), finDelMundo],
      [exc(historia), horas(1), carcelDeReincidentes],
      [exc(parqueNacional), horas(2), tierraDelFuego],
      [exc(lago), horas(1), escondido],
      [exc(lago), horas(1), fagnano]]))),
(viaje(avion, aire), null)])
```

Notice that in the last element of the travel list, the stay is null representing the return travel.

12.4.2 Destination ontology

The *T-Agent* needs to have information about the country and the different possibilities its places bring about. Usually the packages have little information about the destinations and the resources available in them. This domain knowledge is complementary to the package information and very important to infer whether a trip including certain destinations can satisfy some tourist preferences (e.g. natural resources). To structure the knowledge about Argentinian tourism, we analyzed different tourism ontologies and most of them were focused on destinations (see e.g. [121]) including the resources they have, the activities they offer, etc. Inspired in them, the following features were extracted for defining the destination ontology in our prototype.

Destination	::=	*(Name, Coordinates, Zone, [NaturalResource], [ArtificialResource], [Activity])*
Coordinates	::=	*(X, Y)*
NaturalResource	::=	*Resource*
ArtificialResource	::=	*Resource*
Resource	::=	*(KindOfResource, Name)*

The information of almost fifty Argentinian destinations (i.e. all the destinations related to the packages used) has been introduced to fill in this ontology. This information has been extracted from official web-sites.

We use as *coordinates* the geographic coordinates provided by the Instituto Geográfico Militar de la República Argentina.[3] The geographical *zone* assigned to each destination corresponds to the partition of Argentinian provinces into zones proposed by the Secretaría de Turismo de la República Argentina.[4]

An example of the destination structure for the *Ushuaia* city is presented below:

```
localidad(nombre(ushuaia), provincia(tierraDelFuego),
  gps(54.80, 68.31), zona(patagonia),
  naturaleza([(parqueNacional,tierraDelFuego), (canal,beagle)
    , (bahia,lapatala), (lago,roca), (lago,fagnano)
    , (lago,elEscondido), (laguna,negra), (rio,grande)]),
  infraestructura([(museo,finDelMundo), (museo,regional)
    , (museo,acatushun), (historia,presidio)
    , (ingenieria,trenFinDelMundo)],
  actividades([avistajeFauna,esqui,navegacion,pesca,trekking]))
```

The ontology used in this prototype was directly code in a prolog file, but it is

[3] http://www.geoargentina.com.ar
[4] http://www.turismo.gov.ar

possible for the *T-Agent* to receive an ontology built using an ontology editor (via XML code).

12.4.3 Special Relations in the domain

As already mentioned in Section 12.3.2, to increase the domain knowledge of the *T-Agent*, some special relational predicates have been included in the BC language. This allows to encode knowledge about related concepts that makes it possible for the *T-Agent* to expand the search to other terms related to the ones expressed in the tourist's preferences and are used in the selection of the best packages for the tourist. In this implementation we have considered important to include two kinds of relations:

1. ***"to be similar to" relation:*** The BC includes a set of instances of the *similar* predicate on pairs of synonymous or similar concepts according to the tourism domain, composing a so-called similarity dictionary. As the *T-Agent* belief context deals with graded information, these instances may include a degree $g \in [0, 1]$ expressing a sort of semantical distance between terms. The formulae in this dictionary are structured as:

   $$belU(similar(term_1, term_2), g)$$

   For instance, we show a fragment of this similarity dictionary:

   ```
   % accomodation category
   belU(similar(apart, hotel3), 0.75)
   belU(similar(camping, campamento), 1.0)

   % transport category
   belU(similar(bus, colectivo), 1.0)
   belU(similar(bus, trafic), 0.9)

   % nature category
   belU(similar(lago, embalse), 0.7)
   belU(similar(montaa, serro), 0.8)
   ```

2. ***"Better than" relation:*** For the accommodation concepts a "better than" relation has been added to express whether an accommodation is better than another one. This transitive relation allows the *T-Agent* to expand the search of the packages that satisfy the user's preferences, to those that include accommodations better than the selected one. Two formulae expressing these relations and the transitive rule in the BD are the following:

   ```
   belU(mejorQue(hotel5, hotel4)).
   belU(mejorQue(hotel4, hotel3)).
   belU(mejorQue(X, Y)):- belU(mejorQue(X, Z)),belU(mejorQue(Z, Y)).
   ```

12.4.4 Beliefs on desires fulfillment

The T-Agent needs to compute in which degree a particular desire is believed to be fulfilled after a plan execution. This means to compute the degree r of the formula $B([\alpha_P]D)$, where α_P is a tourism package and D is a desire (elementary or combined). This belief degree r is necessary for the agent to estimate the *expected satisfaction* $E(D, \alpha)$ of a desire D by a plan α_P, as we will see later in Section 12.6.1 where this expectation is estimated by the value $E = r \cdot d$, where r is the degree of $B([\alpha_P]D)$ and d the degree of desire D. Notice that, following the model presented in Chapter 5, the truth degree of $B([\alpha_P]D)$ may be considered as the probability of making D true after following plan α_P. In the following we describe how such a probability is estimated from according to the different types of desire types and plans.

Basically, a tourist plan may be considered as a temporal sequence of subplans and the global satisfaction depends on how user's preferences are expected to be satisfied through each stage of the plan trip. As it was presented above, the packages are structured as:

$$Package ::= (Id, Provider, Cost, Trip)$$

where $Trip$ is a travel-stay sequence $[(Travel_i, Stay_i)]$, $i = 1, \ldots, n$. In our approach each pair $(Travel_i, Stay_i)$ is considered as an atomic package stage (subplan), amenable to satisfy some desires. Packages α_P are therefore modelled as composed plans, $\alpha_P = \alpha_1; \ldots; \alpha_n$, alternating travel and stay sub-plans.

Then, the expected satisfaction $E(D, \alpha_P)$ of a desire $D = D_1 \wedge ... \wedge D_n$ through the execution of the plan α_P is computed in our model (see Section 12.6.1) from the expected satisfactions values $E_{ij} = E(D_j, \alpha_i)$ of the elementary desires D_j by the execution of the elementary sub-plans α_i. In turn, to compute each of the E_{ij}'s, the belief degree r_{ij} of achieving the desire D_j through the subplan α_i execution (corresponding to the degree of the formula $B([\alpha_i]D_j)$) is needed. This is described next.

The case of elementary desires

For evaluating the belief degree r in which a package α_P will fulfill an elementary desire D, the agent focuses on either the travel stages or the stay stages in the α_P depending on the kind of desire D specifies. For example, if D is about transport then, only the travel stages in α_P are considered, while if D is related to a natural resource then only the stay stages of α_P are considered. In any case, the belief degree is computed using a set of rules that depend on the kind of desire and on the user's priority criterion.

For example, the BC has a rule setting that "*if the desire D is about accommodation of category c and $stay_i$ of package α_P (i.e. the subplan α_i) offers an accommodation better or equal than c, then the belief degree of fulfilling the desire D*

by subplan α_i *is* $r_i = 1$". In other words, in case D and α_i satisfy these conditions, such a rule would create the formula $(B([\alpha_i]D, 1)$ in the agent's BC theory.

When the tourist's desire D is related to a destination resource (e.g. natural resources, activity) the belief degree of fulfilling it by a plan execution has another interesting characteristic. We have noticed that packages have usually limited information about destinations and their resources. Thus, for belief estimation purposes, besides the package information, the *T-Agent* may need further knowledge about destinations. In our prototype this information is structured in a destination ontology. This amounts to extend the computation of the degree r_i of $B([\alpha_i]D)$ to a *package-destinations* cross inference to assess the fulfillment of the tourist's selected preference in a certain destination, using not only the package supplied information but also the available information about the destination. Therefore, the strategy which is followed is, for each package stage α_i, to evaluate the probability of α_i providing a certain resource D both from explicit information offered in the package (r_{Pi}) and from information inferred from the destination ontology (r_{Oi}). Finally, the T-Agent takes as degree r_i the maximum of both estimations, i.e. $r_i = \max\{r_{Pi}, r_{Oi}\}$.

Combined desires

The DC theory includes conjunction of positive desires. To evaluate the probability of fulfilling the conjunction of elementary desires (e.g. $(D_1 \wedge D_2)$) by the execution of a package α, we assumed that, as random variables, the elementary desires are stochastically independent. Then, from the degrees r_1 and r_2 corresponding to the elementary desires D_1 and D_2 respectively, we can compute the belief degree in achieving their conjunction by executing the plan α using the following rule:

$$\frac{(B[\alpha]D_1, r_1), (B[\alpha]D_2, r_2)}{(B[\alpha](D_1 \wedge D_2), r_1 \cdot r_2)}$$

For example, consider the T-Agent has the following combined desire D specified in the DC:

$$desU(yLst([(zone, patagonia), (activity, rafting)]), 0.8)$$

and the agent has also in her belief context BC the belief degrees of obtaining the elementary desires by a package α, which are respectively:

$$(B[\alpha]patagonia, 1.0) \text{ and } (B[\alpha]rafting, 0.7)$$

Following the rule given above, the T-Agent computes that the belief degree for the combined desire is:

$$(B[\alpha](patagonia \wedge rafting), 0.7)$$

12.5 Planner Context

The Planner Context (PC) is fundamental for the *T-Agent* implementation. The PC unit is assumed to contain a set of available plans, coded as instances of the predicate *planner* with *paq* formulae (see below). The Planner context is responsible for looking among them for *feasible packages*. By *feasible package* we mean a package that fulfills, to some degree, one of the positive desires (elementary or combined) and avoids, as post-condition, the satisfaction of the agent's negative desires above to a given threshold $UmbralN$. The set of feasible plans is determined within this context using an appropriate searching method that takes into account information injected by bridge rules from the BC and DC units, including positive and negative desires, information about packages (including their cost), the agent's beliefs about package destinations and the estimation of the agent's desires fulfillment by the different plan executions. The following forward rule encodes this in the Planner context.

```
des(yLst(DeseosP), _), des(nLst(DeseosN), UmbralN),
planner(paq(IdPaq, Proveedor, Costo, _Recorrido)),
bel(contiene(IdPaq, DeseosP), R),
bel(not(contiene(IdPaq, DeseosN)), UmbralN),
bel(costoNormalizado(Costo, CN), 1)
--:
planner(paqSi(IdPaq, Proveedor, CN, DeseosP), R)
```

For each feasible package, with identifier $IdPaq$, this rule creates into the PC theory an instance of the *planner* predicate with a *paqSi* formula with identifier $IdPaq$. Note that in each instance of a feasible package, its normalized cost ($CN \in [0, 1]$) is used instead of the actual cost. A rule computes this normalized cost as $CN = (Costo/CostoMax)$ where $CostoMax$ is the maximum cost of all the costs of feasible packages.

After the PC has identified the set of feasible packages, they are passed to the Intention context, which is in charge of ranking of these packages according to the user's preferences.

12.6 Intention Context

In order to rank the feasible packages to be offered to the user, the Intention context IC of the *T-Agent* is in charge of estimating the intention degree for each feasible package as a trade off between the benefit (expected satisfaction) and the cost of reaching the user's desires through that package. Thus, first, this context estimates the expected satisfaction $E(D, \alpha)$ of a tourist's desire D assuming she selects a package α. Second, using a suitable bridge rule, it computes the intention degree (the truth degree of the formula $I_\alpha D$) towards the desire D by executing a tourist package α using a function that combines the expected satisfaction $E(D, \alpha)$ and the

normalized package cost CN. In the following Subsections we give some insights of how this estimations are implemented in the *T-Agent*.

12.6.1 Estimating the expected satisfaction of desires

For estimating the expected satisfaction of a tourist's desire D assuming she selects a package α_P, the underlying idea is to consider that each plan α makes D a binary random variable, with a probability distribution

$$Prob_\alpha(D = true) = r, \quad Prob_\alpha(D = false) = 1 - r$$

Now, if d is the user's positive desire degree for making D true, and assuming the positive desire of making D false is 0, then the user's expected satisfaction degree by achieving D through the plan α is clearly

$$E(D, \alpha) = r \cdot d + (1 - r) \cdot 0 = r \cdot d$$

Therefore, to estimate the value $E(D, \alpha)$ one needs to estimate both the probability r of achieving D by α and the (positive) desire degree d for D. These values can be directly obtained from the BC and DC contexts respectively for atomic package components and elementary desires. Indeed, if the plan α consists of a sequence of travel-stay components $\alpha = \alpha_1; \ldots; \alpha_k$ and the desire D is a conjunction of elementary desires $D = D_1 \wedge \ldots D_n$, then the agent contains in her contexts:

BC: instances $(B[\alpha_i]D_j, r_{ij})$, for $i = 1, k;\ j = 1, n$ and

DC: instances (D^+D_j, d_j), for $j = 1, n$, and (D^+D, d) where
$d = 1 - \Pi_{j=1,n}(1 - d_j)$ (see DC context in Section 12.3.2).

and hence, for each i, j we can estimate $E(D_j, \alpha_i)$ with the values $r_{ij} \cdot d_j$. Then, in order to come up with a estimated value for $E(D, \alpha)$, we follow the following steps:

(i) $E(D_j, \alpha)$ is computed for each elementary desire D_j, $j = 1, n$ and then

(ii) $E(D, \alpha)$ is estimated for the combined desire D.

Next we give some insights of how the estimation is done first, for elementary and then, for combined positive desires.

(i) Elementary desires

The expected satisfaction $E(D_j, \alpha)$ of an elementary desire D_j through the execution of the plan α may be computed from the expected satisfactions $E_{ij} = E(D_j, \alpha_i)$ by the execution of its sub-plans α_i, using an appropriate aggregation operator $\oplus$, i.e.:

$$E(D_j, \alpha) = \oplus(E(D_j, \alpha_1), \ldots, E(D_j, \alpha_k))$$

For computing each E_{ij}, the agent uses the probabilities r_{ij} and desire degrees d_j injected from the BC and the DC respectively by a bridge rule. A set of rules play the aggregation role to obtain the expected satisfaction. These rules depend on the kind of desire.

The underlying idea to compute the expected satisfaction of a user's desire D_j (with degree d_j) is to consider the proportion (in terms of duration) of the package components where D_j is expected to be satisfied with respect to the whole trip proposed by the package. Furthermore, the estimation of how much a component of a package α_P (a sub-plan α_i) satisfies a preference may be also graded and is computed depending on what the offer of the sub-plan is, as it is explained next: In our approach we consider for this estimation the relationship of the tourist's desire D_j with the actually proposed in the package D'_j:

- if D'_j is exactly D_j or "better than" than D_j, the expected satisfaction of D_j by the package component α_i is taken as $E(D_j, \alpha_i) = r_{ij} \cdot d_j$, where r_{ij} is the belief degree of $B([\alpha_i]D_j)$.
- if D'_j is similar to D_j to the degree s (see previous subsection 12.3.2), the expected satisfaction of D_j by the package component α_i is taken as $E(D_j, \alpha_i) = r'_{ij} \cdot d \cdot s$, where r'_{ij} is the belief degree of $B([\alpha_i]D'_j)$.

Then, if the package α is composed by different stages, i.e. $\alpha = \alpha_1; \ldots; \alpha_n$, the general way of computing the expected satisfaction $E(D_j, \alpha)$ of the desire D_j by the package α, is defined as

$$E(D_j, \alpha) \;=\; \frac{\sum_i E(D_j, \alpha_i) \cdot Time_{\alpha_i}}{TotalTime},$$

where $Time_{\alpha_i}$ and $TotalTime$ are computed according to the kind of desire D_j.

For instance, if D_j is about accommodation, $Time_{\alpha_i}$ denotes the duration (in days) of the stay α_i and $TotalTime$ is the total duration of the trip. On the other hand, in the case of D being an activity, if the user's preferences specify to do the activity with *high frequency* (see Section 12.2), $Time_{\alpha_i}$ is the duration (in hours) of the activity programmed by α_i and $TotalTime$ is an estimation of the total number of hours the activity could take during the whole trip.

Example: Let us assume a tourist has an accommodation preference of *Apart-Hotel* represented by the desire D:

$$desU(y(comodidad, apart), 0.7).$$

Using the similarity relation between this type of accommodation and a 3 star hotel represented by the instance

$$belU(similar(apart, hotel3), 0.75)$$

the T-Agent considers the tourist will also be satisfied to some degree if he is accommodated in a *3-star Hotel*, i.e. he is assumed to also have the desire D' represented by:

$$desU(y(comodidad, hotel3*),\ 0.7 \cdot 0.75)$$

The *T-Agent* evaluates the expected satisfaction of accommodation through the package *holCalafatePatagonia*. This package has two stay components: 3 days in Calafate with *Apart-Hotel* accommodation and 4 days in Ushuaia with a *3-star Hotel* accommodation (see for details subsection 12.4.1). Considering that in the BC were computed $r_1 = 1$ and $r'_2 = 1$ then, the expected satisfaction of each one of these two package components are respectively as follows:
- $E_1 = r_1 \cdot d = 1 \cdot 0.7 = 0.7$ and
- $E_2 = r'_2 \cdot d' = 1 \cdot 0.525 = 0.525$

Finally, the expected satisfaction of the accommodation desire by the package *holCalafatePatagonia* is computed as the average of the expected satisfactions E_1 and E_2, considering the duration in days of each stay, i.e.:

$$E = \frac{(E_1 \cdot 3) + (E_2 \cdot 4)}{7} = 0.6$$

(ii) Combined desires

If the agent has a combined desire $D = D_1 \wedge ... \wedge D_n$ with degree d (i.e. the formula (D^+D, d) is in the DC) and she has selected a plan α, for each desire D_j the agent can compute the expected satisfaction $E(D_j, \alpha)$ as it was shown in item (i). Then, the agent can estimate the probabilities:

- $Prob_\alpha(D_j = true) = E(D_j, \alpha)/d_j$, for each desire D_j and
- $Prob_\alpha(D = true) = \Pi_{i=1,n} Prob_\alpha(D_j)$.

Finally, she can estimate the expected satisfaction of the combined desire as follows:
- $E(D, \alpha) = Prob_\alpha(D) \cdot d$.

The expected satisfaction is then coded in the intention context following the schema proposed, and using the information introduced by suitable bridge rules from PC (feasible packages), DC (set of desires), BC (priority selected). As we have detailed the estimation of the expected satisfaction depends on the kind of desire (TD) (i.e. preference category and if it is elementary or combined), and is computed by using different aggregation functions ($fes(TD, GD, GR, FU, ES)$) that combines the belief degree of having the desire after a plan execution (GR), the desire degree (GD) and respect the activity frequency preference selected by the user (FU). The rule that compute the espected satisfaction degree (ES) is then coded as follows:

```
planner(paqSi(Id, Proveedor, CN, DeseosP), GR),
des(yLst(D), GD),
bel(frecuencia(FU), 1),
fes(TD, GD, GR, FU, ES)
--:  int(es(D,Id), ES)
```

12.6.2 Computing the intention degrees

After the T-Agent has estimated the tourist's expected satisfaction $E(D, \alpha)$ for each desire D and feasible package α, as a suitable aggregation of the desire degrees d (GD) and the belief degree in achieving the desire r (GR), the corresponding intention degrees are computed in the IC. Namely, for each desire D and a feasible package α to achieve D, the following bridge rule is used to infer a degree for the formula $I_\alpha(D)$:

$$\frac{PC : fplan(D, \alpha, P, A, c), IC : (E(D, \alpha), e)}{IC : (I_\alpha D, f(e, c))} \tag{12.1}$$

This degree is computed, by means of a suitable function f combining the expected satisfaction of the desire through the plan execution (e, which in turn is function of r and d) and the cost of the plan (c). Different functions can model different individual agent behaviors. In the *T-Agent* this function is defined as a weighted average:

$$f(e, c) = \frac{w_{es} \cdot e + w_{cost} \cdot (1 - c)}{w_{es} + w_{cost}} \tag{12.2}$$

where w_{es} and w_{cost} are weights which are set by the *T-Agent* according to the priority criterion selected by the user (minimum cost, preference satisfaction). If the selected priority option is minimum cost then, w_c is set greater than w_{es} and if it is preference satisfaction, w_{es} is given a greater value.

The following bridge rule code infers the intention formulae related to the package Id with the corresponding intention degree (G):

```
planner(paqSi(Id, Proveedor, CN, DeseosP), GR),
int(es(D,Id), ES),
bel(prioridad(PU), 1),
f(ES, CN, PU, G)
--:  int(paqRecomendado(Id), G)
```

After the bridge rule has been applied to all the feasible plans, the IC has in its theory a set of graded intention formulae. The intention degrees are used by the *T-Agent* to rank the feasible packages that communicates to the CC. We opted to select the first nine packages to recommend the tourist.

Finally, the selected packages are passed to the CC unit and then, through the user interface the *T-Agent* outputs to the user the ranking as the system recommendation. For instance, Figure 4 shows a tourist's preference selection and the resulting recommended ranking is shown in Figure 5. After analyzing the recommended packages, the user is prompted by the system to provide his feedback.

12.7 Conclusions

A prototype of multiagent Tourism Recommender system has been implemented. A multiagent approach is suitable for this kind of systems dealing with heterogeneous and distributed information. Particularly we used a g-BDI architecture for modelling the *T-Agent*, showing in this way, that this model is useful to develop concrete agents in real domains.

We remark that the many-valued model of information representation and reasoning in the g-BDI agent, has many advantages for this implementation. First, this model enables an expressive representation of the domain knowledge (agent beliefs), the user's preferences (desires) and the resulting intentions. Secondly, the implemented approach allows the agent to expand the retrieval of feasible packages using similarity relations and domain knowledge, not explicitly included in the package information. Also, the treatment of many-valued information makes it possible to compute in a graded way the expected satisfaction of the different tourist's preferences, by the execution of diverse packages. Finally, the intention degree of a plan towards a desire satisfaction may be computed as a function of diverse factors (e.g. satisfaction, cost, trust). As we can obtain diverse agent behaviors defining different functions for intention computation, these become a crucial point in the agent model.

The implementation of this recommender system and particular of the *T-Agent*, modelled as a g-BDI agent enable as to make some experimentation to validate the system and also to compare our graded BDI model of agent with a simulated crisp BDI version of the *T-Agent*. This experimentation is presented in the following chapter.

Part IV

Experimentation and Discussions

A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it.

A. Einstein

Chapter 13

Recommender System Validation and Experimentation

13.1 Introduction

It has been shown in previous Chapters (11 and 12) that the g-BDI model is useful to design and implement recommender systems. In this Chapter we present the validation of the Tourism Recommender system where its main recommender agent, the Travel Assistant Agent (*T-Agent*), was modelled using our graded BDI approach. The experimentation on the case study aims at proving that this agent model is useful to implement different and rich behaviors. Besides, we show that the results obtained by recommender agents using graded attitudes improve those achieved by agents using non-graded attitudes.

The validation process checks whether the Tourism recommender system designed using this agent model is useful and returns suitable recommendations. As the process of information classification is generally a complex and personal task, we measure the average system behavior over a given population.

We follow two experimental directions. First, we have performed sensitivity analysis to show how the g-BDI agent model can be tuned to define concrete agents having different behaviors, by modifying some of its component elements. For this purpose, in *Experiment 1* we use a Travel agent, called *T2-Agent* which differs from the previously introduced *T-Agent* in the desire context. Later on, in *Experiment 2* we implement an agent, called *T3-Agent* that differs from *T-Agent* in the bridge rule used to obtain the resulting intention degrees.

Second, we describe some experiments we have done in order to compare the performance of recommender agents using the g-BDI model with agents without graded attitudes. Starting from the *T-Agent* and the *T2-Agent* we simulate two families of non-graded BDI recommenders. The way of doing this was by introducing some thresholds for belief and desire degrees. We experiment with the same cases collected in the validation process running them in these agent families. This experimentation compares the performance of the *T-Agent* and *T2-Agent*, both im-

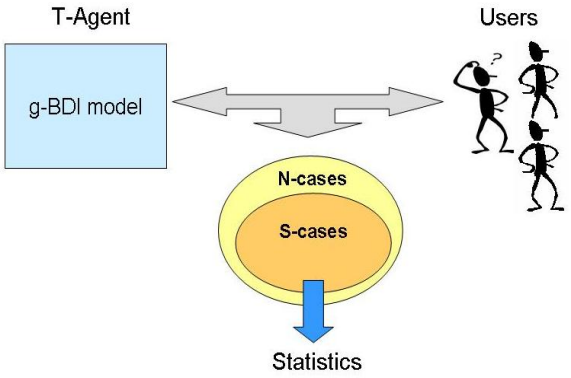

Figure 13.1: The validation process for the Tourism Recommender System

plemented with the g-BDI agent model, with a family of simulated non-graded BDI ones. Finally, we present some statistics and conclusions.

13.2 Validation

In the design and implementation stages of a Recommender System (see Chapter 12) we showed that the g-BDI model of agent is useful to model this kind of systems. In this section we try to answer the question of whether the resulting system is good as a tourist recommender.

Since the *T-Agent* is a personal recommender agent, to analyze its behavior the user's opinion is key. This recommender system is accessible via Internet[1] allowing an online and a multiuser access. The user's opinion is gathered after he receives a ranking of tourist packages. We want to study whether the *T-Agent* is a personal agent satisfying, to some degree, a set of different users. This validation process is illustrated in Figure 13.1 and is explained below.

We use the implementation of the *T-Agent* modelled as a g-BDI agent as detailed in Chapter 12. A set of 40 tourism packages are offered to the *T-Agent* by the provider agents. We have collected near 70 queries from which we have selected a set of 52 complete queries (including user's feedback) made by at least 35 different users, most of them students of a Computer Sciences Department.[2] The preferences and restrictions of the users, together with the system results and the users' feedbacks, constitute our *N-cases* set. Thus, each case in the dataset is composed by:

- **User's profile**: a user ID and his graded preferences and restrictions.
- **Agent result**: the agent returns a ranking of at most k packages.[3]

[1] http://musje.iiia.csic.es/eric/

[2] Departamento de Cs. de la Computacin, Facultad de Cs. Exactas, Ingeniera y Agrimensura, Universidad Nacional de Rosario, Argentina.

[3] the number of ranked packages is a *T-Agent* parameter, experiments are reported for k=9.

ID	NAME	ZONE	gr	NATURAL	gr	INFRAEST.	gr	TRANS.	gr	ACOMOD.	gr	ACTIVITY	gr	COST	DIST.	DAYS	PRIORITY
Id1	Agustín	norte	0,7	montana	1	null	0	null	0	hotel4	0,9	null	0	2000	0	7	satisfaccion
Id2	Martín	centro	0,8	mar	1	espectaculo	0,8	bus	0,9	hotel1	0,7	navegacion	0,7	0	0	10	min_costo
Id3	Paula	patagonia	0,2	lago	0,3	null	0	null	0	null	0	montanismo	0,1	0	0	10	satisfaccion
Id4	Verónica	centro	0,5	mar	0,5	espectaculo	0,5	bus	0,5	hotel3	0,5	null	0	3000	0	7	satisfaccion
Id5	Pablo	centro	0,5	mar	1	null	0	null	0	null	0	null	0	0	0	7	satisfaccion
Id6	camilo	patagonia	0,9	lago	0,5	null	0	bus	0,8	camping	0,8	navegacion	0,5	0	0	15	min_costo
Id7	Cecilia	centro	0,2	mar	1	museo_arqueo	0,1	bus	0,8	hosteria	0,8	navegacion	0,5	800	2000	15	satisfaccion

ID	R1	R2	R3	...	R9	Feedback1	Feedback2	Feedback3
Id1	avaMinaClavero	avaLaCumbre	avaLosCocos		holBariloche7Lagos		(expMendoza, avaLaCumbre, avaMinaClavero)	
Id2	tatMarDelPlata	expBuenosAires	holIbera		tatLasGrutas	ok		
Id3	tatBariloche	holEsquel	holBariloche7Lagos		avaConcordia	ok		
Id4	tatMarDelPlata	expBuenosAires	holBsAsTangoExpress		avaVillaGralBelgrano	ok		
Id5	tatMarDelPlata	tatLasGrutas	expValdes		avaLosCocos	ok		
Id6	tatLasGrutas	tatBariloche	avaFederacion		holBariloche7Lagos		(tatBariloche, holBariloche7Lagos, holEsquel)	
Id7	tatMarDelPlata	tatVillaGesell	tatLasGrutas		avaLosCocos			Comentario

Figure 13.2: Example of records containing some user profiles (above) and a subset of the *T-Agent* ranking together with the corresponding user's feedback (below)

- **User's feedback**: after analyzing the information of the offered plans the user gives a feedback that may be:

  1. *Correct order*: if the tourist considers that the order of the first three packages is correct (called Feedback1 in table of Figure 13.2).
  2. *Different order*: if the tourist finds what he wants in the ranked list but considers that the first three packages are not in the right order. In this latter case, the user gives his most preferred order (called Feedback2).
  3. *Incorrect*: the user does not agree with the recommendation (called Feedback3).

  In this validation process we consider that when the user's feedback is (1) or (2), this corresponds to a satisfactory agent result, as he can find what he want among the recommended options.

Actually, in the validation process, we have only taken into account those cases which included user's feedback. Examples of some records used and their most significant fields are shown in the tables of the Figure 13.2.

Results:

From the selected 52 cases (*N-cases*) we have separated those having a satisfactory feedback (i.e. *Correct* or *Different order*) from the unsatisfactory ones. The cases where the user gives his own ranking (*Different order*), are indeed very valuable because it means that the user took time to analyze the offers proposed by the

system, while cases with Feedback 1 (*Correct order*) may sometimes correspond to a "quick answer".

The *N-cases* are classified by their different feedback categories in Table 13.1. From these results, the global behavior of the *T-Agent* may be considered useful in most cases (73% of *N-cases*). These are preliminary results and the recommender systems may be improved to obtain better performance if we want it to be more qualified.

Consults (*N-cases*)	Correct order	Different order	Incorrect	Satisfactory (*S-cases*)
100%	40.4%	32.7%	26.9%	73.1%

Table 13.1: Result of the *N-cases* in the validation process

In order to give a general measure of the *T-Agent* results over the satisfactory cases (*S-cases*), we have evaluated how close is the *T-agent* ranking respect to the user's own ranking. For this, we choose the block (Manhathan) distance between the position of the first three packages selected by the user and their position in the system ranking. This distance was adopted because it is appropriate for capturing positional differences.

Namely, assume the user's feedback is $U_i = (P_{i1}, P_{i2}, P_{i3})$ and the *T-Agent* ranking for this query is $R_i = (R_1, R_2, ..., R_9)$. Then, if $P_{i1} = R_j$, $P_{i2} = R_k$, $P_{i3} = R_n$, the distance between the user's and the system rankings is defined by:

$$Dist\,(U_i, R_i) = |1 - j| + |2 - k| + |3 - n|$$

As for example consider the first case showed in Figure 13.2 where the *T-Agent* ranking of packages is as follows:

R_i= *(ayaMinaClavero, ayaLaCumbre, ayaLosCocos, ayaVillaGralBelgrano, ayaVillaCarlosPaz, expMendoza, holMendozaVinos, holEsquel, holBariloche7Lagos)*

and the user's own ranking is:

U_i=*(expMendoza, ayaLaCumbre, ayaMinaClavero)*

then, the distance for this case is:

$$Dist\,(U_i, R_i) = |1 - 6| + |2 - 2| + |3 - 1| = 7$$

Notice that for the cases with *Correct order* feedback (Feedback 1) the distance is 0 and in the worst case this distance takes the value of 18. The table in Figure 13.3 (left) shows the block distances for all the cases with satisfactory user's feedback (*S-cases*) and the distance frequencies for *S-cases* can be seen in Figure 13.3 (right).

Case	Distance	Case	Distance
Id1	4	Id29	0
Id3	4	Id30	7
Id4	6	Id31	0
Id5	0	Id32	0
Id6	6	Id34	0
Id7	2	Id35	0
Id9	2	Id36	11
Id10	0	Id38	1
Id11	0	Id39	0
Id13	0	Id40	0
Id14	0	Id41	0
Id15	3	Id42	0
Id18	11	Id44	0
Id19	3	Id45	0
Id20	0	Id47	3
Id21	7	Id48	14
Id22	17	Id49	0
Id27	0	Id50	0
Id28	0	Id52	11
Average Distance			2.95

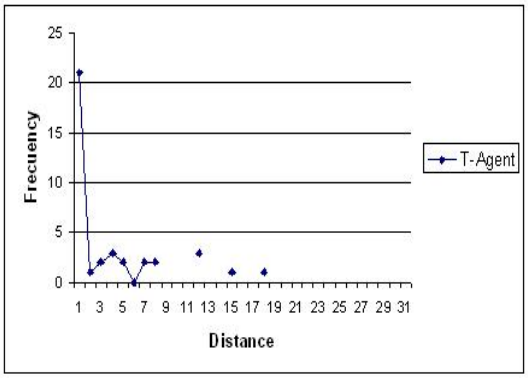

Figure 13.3: Distances of the *T-Agent* over the *S-cases* (left) and the corresponding frequencies (right).

We analyzed the incorrect cases and the comments attached (if any) about the user dissatisfaction with respect to the system recommendation and they were somewhat scattered. Apart from that, in some of these incorrect cases we detected a system shortcoming related to the tourism knowledge base, the destination ontology used for this experimentation was incomplete with respect to the popular knowledge. We found another limitation due to the system interface that was built for experimental use and need more detailed explanations about the different items required to the users, to avoid misunderstandings.

Therefore, we believe the *T-Agent* behavior may be improved by completing these ontologies and refining the user interface.

Finally, the *S-cases* (see table in Figure 13.3) yield an average distance of 2.95 in the scale $[0, 18]$, and hence giving a good global measure result. Summarizing, we have obtained preliminary satisfactory results of the Recommender System in this validation process, that allows us to claim that "the *T-Agent* recommended rankings over Tourism packages are in most cases near to the user's own rankings". Even thought we recognize that future work is needed to improved the system in the pointed directions, we consider that these first results on the N-cases allow as to focus on the experimentation of our agent model using this Recommender.

13.3 Experimentation

In this section we present the experimentation we have made in two directions. The first one, a Sensitivity analysis, has the purpose of finding out how much the general g-BDI agent architecture can model different behaviors by varying some of its components. The second one, aims at checking whether the distinctive feature of the g-BDI agent model, which is the gradual nature of mental attitudes, actually makes a difference (in terms of better results) with simulated BDI non-graded models.

In Figure 13.4 we present the experimentation schema. The Sensitive analysis is based on Experiments 1 (Exp-1) and 2 (Exp-2). Experiments 3 (Exp-3) and 4 (Exp-4) correspond to the comparison between graded agents (g-BDI model) and non-graded ones (c-BDI model).

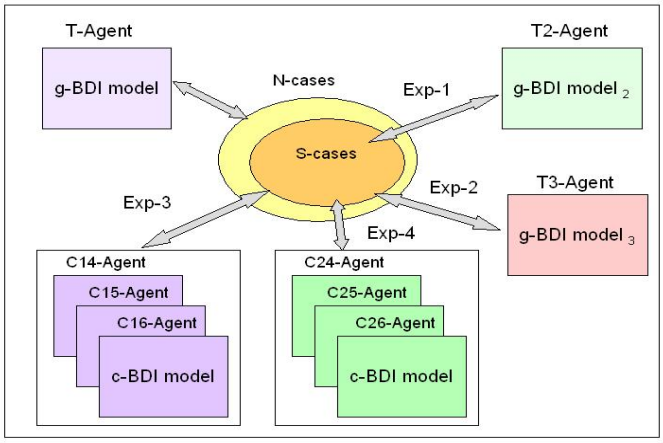

Figure 13.4: Experimentation scheme

13.3.1 Sensitive model analysis

We have performed two experiments to analyze how the overall recommender system behavior can be modified by tuning some of the *T-agent* components. First, in *Experiment 1* we change the theory of one of the mental contexts, the desire context DC, using a different method to compute the desire degree of preference combinations. Then, in *Experiment 2* we modify a bridge rule definition by changing the function used to obtain the intention degree.

Experiment 1

(1) We use the tourism recommender agent *T2-Agent*: this agent was developed changing the desire context of the *T-Agent* (see Section 12.3.2). The modification in this context is related to the way the desire degrees are computed. The underlying idea was to weight not only the preference degrees but also the number of preferences we are considering in each combined desire, as to have a more dispersed distribution of desire degrees, giving relevance to the desires that combine a higher number of preferences. For this purpose, in the Desire Context of the *T2-Agent* we use as degree for desire D the value

$$d' = 1/2 * (d + \frac{CardD}{CardPref})$$

where d is the desire degree used in the *T-Agent* (described in Section 12.3.2), *CardD* is the number of preferences considered in the desire D and *CardPref* is the total number of preferences selected by the user.

For example, if a user's graded preferences (elementary desires) are as follows:

desU(y(zona, patagonia), 0.9)

desU(y(transporte, avion), 0.7)

desU(y(comodidad, hotel3), 0.6)

In the *T-Agent* we compute the degrees for the combined desires as:

desU(yLst([(zona, patagonia), (transporte, avion)]), 0.97),

where $d_1 = 0.97$

desU(yLst([(zona, patagonia), (transporte, avion),

(comodidad, hotel3)]), 0.988), where $d_2 = 0.988$

For the *T2-Agent* we proposed to use d'_1 and d'_2 instead:

$$d'_1 = 1/2 * (0.97 + \frac{2}{3}) = 0.818 \quad \text{and} \quad d'_2 = 1/2 * (0.988 + 1) = 0.994$$

We notice that the difference between these desire degrees in *T2-Agent* are greater than the corresponding one in *T-Agent* (e.g. $d'_2 - d'_1 = 0.176$ and $d_2 - d_1 = 0.018$).

(2) We consider the *S-cases*, the set where the results were satisfactory (see Section 13.2).

(3) The user's preferences of the *S-cases* are run in the *T2-Agent*.

(4) We compare the *T2-Agent* results with the *S-cases user's feedbacks* we have for the *T-Agent* results and compute distances.

Results:

In this experiment we are comparing the ranking proposed by the *T2-Agent* with the first three packages extracted from the *T-Agent* recommendation or the user feedback. Some of these packages may not be found in the *T2-Agent* answer. Again, we use the Block distance to have a global measure of the *T2-Agent* performance. For the missing packages, we take an optimistic approach assuming that the distance is 10 (supposing that the missing package would be the first one immediately after those appearing in the ranking). This approach was decided because we consider this is a kind of "indirect measure" (we are comparing the *T2-Agent* results with the feedbacks the users gave to the *T-Agent* results) and can give us worst results than in a direct one.

Two global measures are computed in this experiment, the average of distances excluding the cases having missing packages in the current ranking and the total average, that includes all the cases. The distance frequencies corresponding to the *T2-Agent* results for all the *S-cases*, including the distance average and the total average, are shown on the Table 13.2 and illustrated in Figure 13.5.

The average of the distances between the *T2-Agent* rankings and those of the *S-cases* is 2.85 and the total average is 4.23. Comparing the first global measure with the one obtained for the *T-Agent* results, we notice that this measure is slightly better than the one computed for the *T-Agent*. We believe that had we had a direct measure of the performance of the *T2-Agent* (comparing the *T2-Agent* ranking with the corresponding user's feedback) we would have obtained better results. We can conclude that the recommenders *T-Agent* and *T2-Agent* have similar behavior, but results of the second one are a little bit closer to the user's rankings.

Experiment 2

The same steps followed in *Experiment 1* were taken but, in this experiment, for item (1) we use a new tourism recommender agent called *T3-Agent*. This new agent was defined from *T-Agent* by changing one of its bridge rules. Namely, we have modified the function that is used by bridge rule 12.1 (described in Chapter 12.6) to compute the intention degree of a package α in order to satisfy a set of user's preferences φ ($I_\alpha\varphi$). We have used for *T3-Agent* a function of the desire degree (d), of the belief degree of satisfying the goal by the plan execution (r) and the cost of the plan (c): $f(d, r, c)$. This function assigns an intention degree according to two different priorities: Preference Satisfaction or Minimum Cost, by following two lexicographic orderings, namely:

- when the *Preference Satisfaction* criterion is selected, we consider to assign as the degree of intention $I_\alpha\varphi$ the 3-tuple $(d', r, 1 - c)$, where d' is the desire degree of φ, r is the belief degree in satisfying the user's preferences φ by the considered plan α, and c is the cost of the plan α. Then, we use the lexicographic order over the product space $[0, 1]^3$ to rank the 3-tuples and hence, the intentions.

- when the *Minimum Cost* criterion is selected, we consider the degree of $I_\alpha\varphi$ to be the 3-tuple $(d', 1 - c, r)$ and then, intentions are again lexicographically ordered.

Results

The distance frequencies of the results of the *T3-Agent* for the *S-cases* are shown in Table 13.2 and are illustrated in Figure 13.5 (and compared with the ones obtained by *T-Agent* and *T2-Agent*). The average of the distances in this case was 4.97, worst than the previous experiments, and the total average is 6.73. This means that the ranking obtained by this *T3-Agent* is farther from the user's ranking obtained in the validation process than the results of the previous versions *T-Agent* and *T2-Agent*.

This fact does not necessarily mean that *T3-Agent* behavior is worst, perhaps it finds different options from these found by the *T-Agent*.

In summary, we can state that the g-BDI model allows us to easily engineer recommender agents showing different behaviors.

Distance		**Frequency**	
	T-Agent	T2-Agent	T3-Agent
0	21	11	5
1	1	4	3
2	2	4	3
3	3	5	3
4	2	1	2
5	0	1	2
6	2	4	3
7	2	2	0
8	0	0	3
9	0	0	2
10	0	1	3
11	3	2	1
12	0	1	4
13	0	1	1
14	1	0	1
17	1	0	0
18	0	0	1
26	0	0	1
30	0	1	0
Average	2.95	2.85	4.97
Tot.Average	2.95	4.23	6.73

Table 13.2: Distance frequencies for *T-Agent*, *T2-Agent* and *T3-Agent*.

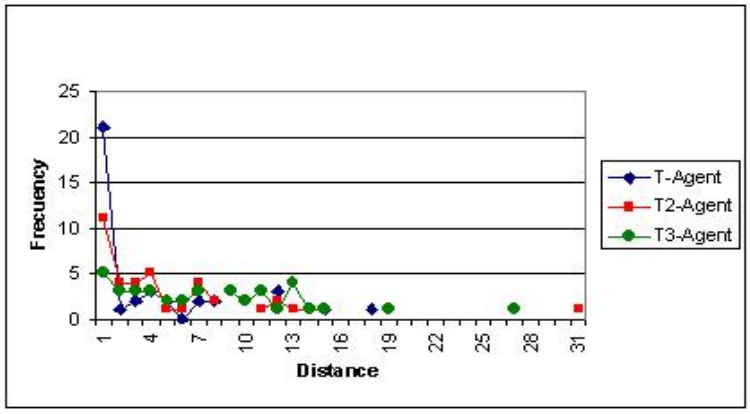

Figure 13.5: Distance frequencies for *T-Agent*, *T2-Agent* and *T3-Agent*.

13.3.2 Graded vs. non-graded model comparison

In these experiments we compare the g-BDI agent model with non-graded (two-valued) BDI architectures. We want to show that the results obtained by our rec-

ommender agents using graded attitudes improve those achieved by agents using non-graded attitudes.

We use the *T-Agent* and *T2-Agent* prototypes as the g-BDI model implementations. Since the development of a Tourism Recommender using a traditional BDI architecture (e.g. dMARS, Jack, Jason or Open-PRS) would be a time-demanding task and since also different factors would possibly interfere in the comparison of the results (e.g. how the agent builds plans, which decision process she uses), we have decided to use simulated non-graded versions of the g-BDI architecture of the tourism agent. Starting from the recommender agents *T-Agent* and *T2-Agent* we keep their multi-context architecture and their logic schemes for contexts.[4] Then, we use some thresholds to make the desire and belief attitudes two-valued (i.e., their degrees will be allowed to only take values in $\{0, 1\}$). The intention degrees have been left many-valued as to obtain a ranking of the selected packages.

Experiment 3

We have followed the same procedure as in previous experiments but, in this case, we define a family of Tourism Recommender agents called *C1j–Agents*, derived from the *T-Agent* and that simulate two-valued models of BDI agents.

Each *C1j–Agent* has been developed by introducing thresholds in the context DC (τ_d) and in the context BC (τ_b) of the *T-Agent*, to decide which formulae in these contexts are considered to hold (i.e. those with degree 1) and which do not (i.e. those with degree 0). Thus, the following internal processes are introduced in these contexts:

- DC: before introducing formulae like $(D^+\phi, d)$ in the DC it is checked whether $d \geq \tau_d$; if so, the formula $(D^+\phi, 1)$ is added in the context, otherwise this desire is discarded (assuming $(D^+\phi, 0)$).
- BC: the same happens when the belief context evaluates the degree r of formulae like $(B[\alpha]\varphi$, r), if $r \geq \tau_b$ then the formula $(B[\alpha]\varphi, 1)$ is added to the BC, otherwise its degree is considered to be 0.

As for the setting of the different thresholds, we analyzed the desire and belief degrees distribution in the *T-Agent* previous executions. Some tourists used a scale [1,5] to give their preferences but most of them used around five different values between [3,9]. This means that most of the desire degrees concentrate in the interval [0.3, 1] (as the degree of conjunctions is greater or equal than their components degrees). With respect to belief degrees, the estimation of the satisfaction by a plan execution, usually took values in the interval [0.5,1]. Given these observations, we experimented with different thresholds (i.e. 0.4, 0.5 and 0.6) that were a good

[4]This is possible as the many-valued frameworks used for the mental contexts are extensions of classical logic used in the two-valued models.

representation of the variations in the agents' results. Thus, we have defined the following "two-valued" BDI agents:

- *C14-Agent* uses $\tau_d = \tau_b = 0.4$
- *C15-Agent* uses $\tau_d = \tau_b = 0.5$
- *C16-Agent* uses $\tau_d = \tau_b = 0.6$

Then, we run the *S-cases* in each agent of this two-valued family and compared the results with the *S-cases* feedback by computing the Block distances. As in previous experiments we have used the distance average and the total average as global measures. The results are shown in Table 13.3.

Experiment 4

We repeat Experiment 3 but using a second family of non-graded Recommender agents, called *C2j-Agents*. To create this family we start from the *T2-Agent* (see Experiment 1). We use the same thresholds and procedure as in Experiment 3, to make the desire and the belief degrees two-valued:

- *C24-Agent* uses $\tau_d = \tau_b = 0.4$
- *C25-Agent* uses $\tau_d = \tau_b = 0.5$
- *C26-Agent* uses $\tau_d = \tau_b = 0.6$

Results

The distance frequencies of the results of these two families of crisp agents deriving from *T-Agent* and *T2-Agent* are respectively shown on Figures 13.6 and 13.7 .

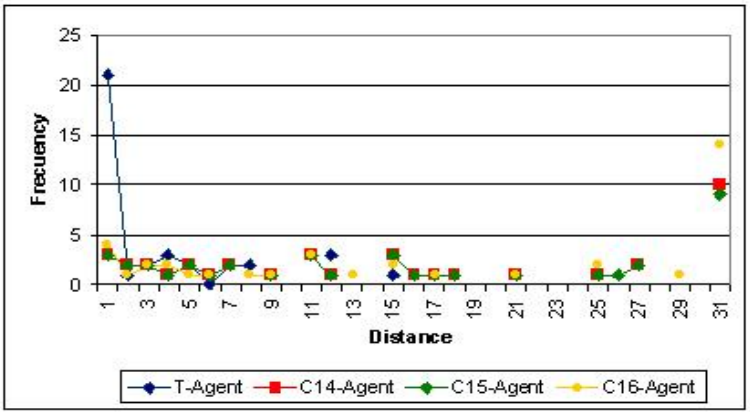

Figure 13.6: Distance frequencies for the *T-Agent* family.

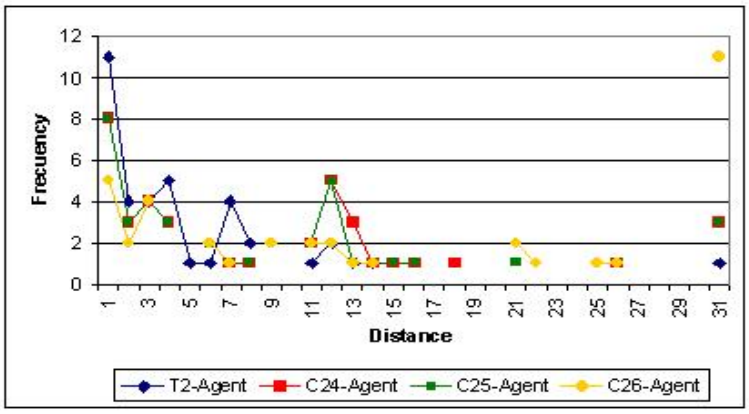

Figure 13.7: Distance frequencies for the *T2-Agent* family.

On the one hand, analyzing the graphic corresponding to the *T-Agent* family (Figure 13.6), we can see that the performance of the *T-Agent* is the best, reaching the maximum of the distance frequencies at 0 and furthermore, almost all the relative maximums are near 0. The behavior of the non-graded agents (i.e. *C14-Agent*, *C15-Agent* and *C16-Agent*) are in general very similar and most of the relative maximums are spread in the interval [0,15]. The absolute maximum move to a distance value of 30, meaning that an important number of queries had feedback packages not found in the corresponding rankings made by the different *C1j-Agents*. In this sense, *C14-Agent* and *C15-Agent* behave better than *C16-Agent*, as their results give lower frequency value at 30.

On the other hand, the distance frequencies graphic of the *T2-Agent* family (Figure 13.7), also shows that the performance of the *T2-Agent* using the graded model is better than the non-graded agents in this family. Besides, in this experiment we obtain similar good results for *C24-Agent* and *C25-Agent* (better than the non-graded ones for the *T-Agent* family). Also, the relative maximum of the frequencies for the *C26-Agent* are very near 0. But in this case, the absolute maximum (reach at a distance frequency of 30) increased its value, meaning that many packages selected by the users are out of this agent ranking.

In these experiments, the distance average and total average are computed as in previous experiments and are gathered in the following table 13.3.

Comparing the averages obtained with the two-valued models of recommenders (deriving from *T-Agent* and *T2-Agent*) we can see that those corresponding to the thresholds 0.4 and 0.5 are very similar. The average achieved with the threshold 0.6 is the best in the *T-Agent* family and is almost the best in the *T2-Agent* one, but the total average is greater, meaning that we have more packages of the *S-cases* feedback out of the system rankings. The number of missing packages, reflected in the total average and in the frequencies of the distance values over 18, is a good indicator of the similarity between the user's ranking and the agent results. In both

	T-Agent family		*T2-Agent* family	
	Average	**Tot. Average**	**Average**	**Tot. Average**
g-BDI model	2.95	2.95	2.85	4.23
$\tau_d = \tau_b = 0.4$	6.43	14.96	4.04	8.36
$\tau_d = \tau_b = 0.5$	6.43	14.83	3.50	8.07
$\tau_d = \tau_b = 0.6$	4	17.41	3.55	14.43

Table 13.3: Distance average results for the *T-Agent* and *T2-Agent* families.

families we can see that these indicators are better for the recommenders using the graded model. Furthermore, we can see that the distance average (excluding the missing packages) of the recommenders using graded models are better than the simulated two-valued ones (using different thresholds).

These results give support to the claim that the recommender agents modelled using graded BDI architectures provide better results than the ones obtained using two-valued BDI models.

13.4 Data analysis

The purpose of this analysis is to investigate which of the differences found in the different experiments we carried out are statistically significant, meaning that the differences observed among the distance averages are significant and hence, the agent behaviours are different.

First, we present some descriptive data resuming the results obtained with the different agents used, namely: *T-Agent*, *T2-Agent* and *T3-Agent* (g-BDI models); *C14-Agent*, *C15-Agent* and *C16-Agent* (*T-Agent* two-valued family); and *C24-Agent*, *C25-Agent* and *C26-Agent* (*T2-Agent* two-valued family). Then, to compare the different results, we apply the Analysis of Variance (ANOVA), we use as the analysis variable the Block distance between the agent results and the *S-cases* feedbacks (described in Section 13.2).

In this analysis we consider all the different agents' results over all the *S-cases* hence, we also take into account the results having missing packages. Then, the number of queries considered for each agent is *S-cases* cardinality (38 cases for this experimentation). For the descriptive analysis we extracted the following information: the total distance average (Tot. average), standard deviation and maximum distance. The minimum distance takes value 0 in all cases. The descriptive information for the different agents is gathered in the Table 13.4.

The total average for the graded agent models *T-Agent*, *T2-Agent* and *T3-Agent* are better than the two-valued families, as it is shown in Figure 13.8.

Next, we apply the Analysis of Variance (ANOVA) for the distances of the different agents' results over the *S-cases* to analyze whether the differences between the total averages obtained by the different agent's results, are significant. The clas-

Agent	Tot. Average	Standard deviation	Maximum
T-Agent	2,95	4,52	17
T2-Agent	4,23	5,74	30
T3-Agent	6,73	5,75	26
C14-Agent	14,97	11,39	30
C15-Agent	14,84	11,24	30
C16-Agent	16,76	12,23	30
C24-Agent	8,37	8,87	30
C25-Agent	8,08	8,95	30
C26-Agent	14,45	12,08	30

Table 13.4: Descriptive measures of the distances obtained over *S-cases*, using different agents.

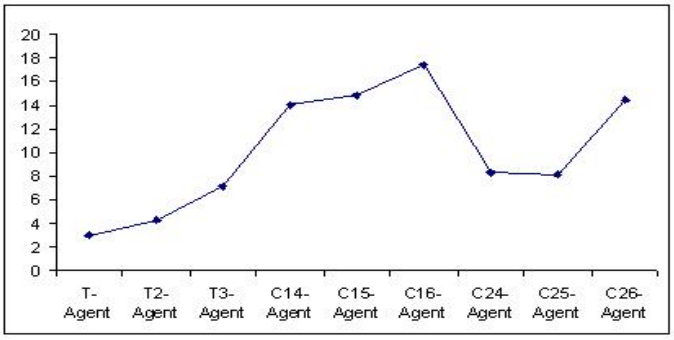

Figure 13.8: Total averages for the different agents.

sic ANOVA includes information about: degrees of freedom (DF), Sums of Squares (SS), Mean Square (MS), Test Statistic (F) and P-value, representing the probability that our hypothesis " the distance averages of the different agents' results are equal", is true. In this analysis we consider our eight different agents whose behaviour we want to compare, and each case in *S-cases* set is consider as a block thus, we have 38 different blocks. The ANOVA results for our case study are presented in Table 13.5.

As the P-value are very small, we found that the total average differences among the agent's results over *S-cases* are significant. Thus, we decided to analyze, using some statistical tool, which of these differences were statistically significant. For this purpose, the Tukey confidence intervals [15] were built for the distance differences between pair of agents' results. For a particular interval, if it is the case that the value 0 is included in the interval, the corresponding difference is not consider statistically significant.

Among all the possible comparisons we consider relevant for our experimentation:

1. the differences between the results of *T-Agent* with the other graded agents: *T2-Agent* and *T3-Agent*, and with its corresponding two-valued family: *C14-*

Source of variation	DF	SS	MS	F	P-value
Agent	8	8110.76	1016.07	19.46	0.000
Block	38	14074.63	370.38	7.09	0.000
Error	295	15401.40	52.21	19,46	
Total	341	37586.79	52.21	19,46	

Table 13.5: ANOVA for the distances between the different agents' results and the S-cases

Agent, *C15-Agent* and *C16-Agent*.

2. the differences between the *T2-Agent* with *T3-Agent*, and with its corresponding two-valued family (*C2j-Agent*).

In the following we describe these comparisons supported by Tukey's intervals:

1. The Figure 13.9 shows the simultaneous Tukey's confidence intervals (95%), for the difference between the *T-Agent* average and the other agents' averages ($\mu_{agent} - \mu_{T-Agent}$) over *S-cases*. Where respectively L_l represents the lower limit of the confidence interval, C the center (is the difference between the distance averages of the corresponding agents' results), and L_u its upper limit.

Agent	L_l	C	L_u
T2-Agent	-3,856	1,289	6,435
T3-Agent	-1,356	3,789	8,935
C14-Agent	6,881	12,026	17,172
C15-Agent	6,749	11,895	17,040
C16-Agent	8,670	13,816	18,961

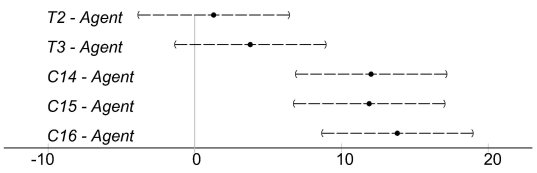

Figure 13.9: Tukey's confidence intervals for the differences between *T-Agent* and other relevant agents ($\mu_{agent} - \mu_{T-Agent}$).

In the cases where the interval does not contain the value 0, the difference between the averages we are comparing are considered statistically significant. As can be seen in Figure 13.9 the results of *T-Agent* does not present significant differences with *T2-Agent* and *T3-Agent* results. Even so, we can notice that the last case is in a limit situation (i.e. L_l is near 0). On the other hand, they clearly present significant differences with *C14-Agent*, *C15-Agent* and *C16-Agent* results.

2. The Figure 2 shows the simultaneous Tukey's confidence intervals (95%), for the average differences between the *T2-Agent* with the *T3-Agent*, and with the corresponding family of two-valued agents (*C2j-Agent*).

   The Tukey's intervals in Figure 2 show that the results of *T2-Agent* do not present significant differences with the *T3-Agent* ones. On the other hand,

Agent	L_l	C	L_u
T3-Agent	-2,646	2,500	7,646
C24-Agent	-1,209	3,976	9,161
C25-Agent	-1,304	3,842	8,988
C26-Agent	5,065	10,211	15,356

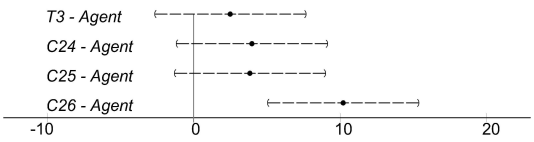

Figure 13.10: Tukey's confidence intervals for the differences between *T2-Agent* and other relevant agents ($\mu_{agent} - \mu_{T2-Agent}$).

they present significant differences with *C26-Agent*. Besides, we found that the interval corresponding to *C24-Agent* and *C25-Agent* differences are in a limit situation (the corresponding L_l are slightly higher than 0). We believe that in the case of considering a greater number of cases, or a lesser confidence percentage in the Tukey's intervals, these results may also present a significant difference.

13.5 Conclusions

In this Chapter we have focussed on the validation and experimentation of g-BDI agents using as a case study a Tourism recommender agent. First, the results of the validation performed allows us to conclude that g-BDI agents are useful to build recommender systems in real and rich domains such as tourism. Second, we have also performed a sensitivity analysis showing that a g-BDI agent architecture can engineer concrete agents having different behaviors by suitably tuning some of its components. The results of a third experiment support our claim that the distinctive feature of recommender systems modelled using g-BDI agents, which is using graded mental attitudes, allows them to provide better results than those obtained by non-graded BDI models. Finally, supported by the analysis of variance we found that even there are any difference among the results given by the different agents modelled using the g-BDI architecture (i.e. *T-Agent*, *T2-Agent* and *T3-Agent*) they are not statistically significant, but the differences between the graded agents *T-Agent* and *T2-Agent* and the families of two-valued agents are significant.

Chapter 14

Discussion

In this final Chapter we present the different contributions of this Thesis. In the different Chapters of this dissertation we have presented partial conclusions and now we gather the most relevant ones. Besides, we present some future lines of work that has been opened during this research work. Finally, we list the related publications to the evolution of this Thesis.

14.1 Contributions of this Thesis

The focus of our research work has been to develop a graded intentional agent architecture. In this direction we have proposed a formal well-grounded agent model capable of representing and reasoning with graded mental attitudes. Besides, our aim was to give this model computational meaning and also to develop a methodology to engineer concrete agents based on this architecture. Thus, we have made contributions to diverse areas related to the core of our Thesis work as preference representation and reasoning, process calculi and agent-based software engineering.

14.1.1 Contributions respect to BDI architectures

The main contribution of this Thesis is the proposal of a general graded BDI agent model. In this model, the agent graded attitudes have an explicit and suitable representation. Belief degrees represent the extent to which the agent believes a formula to be true. Degrees of positive or negative desires allow the agent to set different levels of preference or rejection respectively. Intention degrees also give a preference measure but, in this case, modelling the cost/benefit trade off of achieving an agent's goal. Then, agents having different kinds of behaviour can be modelled on the basis of the representation and interaction of their graded beliefs, desires and intentions.

We consider our Thesis work is an important contribution in the agent architectures field. Our g-BDI architecture is more flexible than classic BDI models and we have shown that is capable to develop agents with improved performance.

Several factors have contributed to the importance of the BDI model in the Agent community. The BDI architecture is one of the best models of practical reasoning that is based on well understood logical foundations. Besides, this model has proved to have the essential components to cope with complex, real world applications.

As we mentioned in Chapter 2 there is a family of BDI architectures. In a wide sense, they are models of practical reasoning that explicitly represent the agent mental attitudes i.e. belief, desire and intentions, perhaps among other attitudes. In the following items we point to the most relevant characteristics that distinguish our g-BDI model with respect to the generalities of the BDI family:

- The g-BDI architecture includes an explicit representation of agent uncertain beliefs, and graded desires and intentions. These graded attitudes are represented by well-founded fuzzy modal logics. Then, diverse uncertainty models can be represented to reason about the different attitudes, by defining suitable modal theories over suitable many-valued logics.

- The g-BDI model is specified using multi-context systems. This specification enables to use different logics in a way that keeps the logics neatly separated. This either makes it possible to increase the representational power of BDI agents —selecting the most suitable logic for each attitude, or to simplify agents conceptually —having the different logics in separate contexts. Then, the MCS specification of g-BDI agents has several advantages from both a software engineering and a logical point of views.

- To represent the agent beliefs, the belief language is built over a dynamic logic language, as to explicitly represent and reason about actions and the changes their executions produce.

- With respect to desires, besides representing graded positive desires our agent model includes the formalization of graded negative desires, to represent respectively the agent desired and rejected states.

- In our approach, an agent intention is considered a pro attitude that results from the agent beliefs and desires, and thus, is not a basic attitude (as proposed in [125]). Intentions, as well as desires, represent another type of agent preferences. However, we consider that intentions cannot depend just on the benefit, or satisfaction, of reaching a goal (represented by desire degrees), but also on the world state and the cost of transforming it into a world where the goal is satisfied. Then, by allowing a graded representation of the strength of intentions we are able to attach to intentions a measure of the cost/benefit relation involved in the agent actions towards the intended goal.

- For the different mental attitudes we have presented a sound and complete basic logic. In the case of beliefs, two logic frameworks have been formalized to represent different uncertainty models. In the case of desires, different logic

schemas were proposed to represent alternative constraints between positive and negative desires, over a formula and its negation.

- In the g-BDI architecture the interrelation among attitudes, as for example different realisms, may be suitably represented by bridge rules (see Section 7.2).

- The g-BDI agent model takes advantage of the agent graded beliefs and desires (both positive and negative) in the agent deliberation process to derive graded intentions. Different agent behaviours can be modelled by combining in different ways these factors. This is tunned by a particular bridge rule.

  The agent performance may be improved by using these graded attitudes as it was shown in the experimentation of a case study (see Chapter 13).

14.1.2 Contributions on related fields

Besides the definition of this novel g-BDI architecture, we have made some additional contributions that are situated on the following fields:

1. **Knowledge representation and reasoning:** *a logical framework with a sound and complete axiomatics for representing desires and intentions, was proposed.*

   Considering the desire representation in our agent model, we based our work on the bipolar model due to Benferhat et al. [12]. We have extended the state of the art by giving a sound and complete axiomatics and defining different logical schemas to represent some additional constraints over preferences. In addition, we have presented a logical system for intentions and we have shown that the framework is expressive enough to describe how desires (either positive or negative), together with other information, can lead agents to intentions.

2. **Process calculi:** *a Multicontext calculus (MCC) to define operational semantics for multi-context systems was developed and we used it for giving semantics to the g-BDI agent model.*

   In order to cope with the operational semantics aspects of the g-BDI agent model, we have first defined a Multi-context calculus (MCC) for Multi-context systems (MCS) execution. The calculus proposed is based on Ambient calculus [28] and includes some elements of the Lightweight Coordination Calculus (LCC) [148]. The operational semantics for this language was given using Natural Semantics.

   We expect that MCC will be able to specify different kinds of MCSs. Particularly, we have shown how graded BDI agents can be mapped into this calculus. Through MCC we have given this agent model computational meaning and in this way, we moved one step closer to the development of an interpreter of the

g-BDI agents. We think that the implementation of agent architectures using process calculi, in particular ambient calculus, would give a uniform framework for agent architectures, multiagent systems and also electronic institutions.

3. **Agent based software engineering:** *a methodology for engineering agent based systems composed by agents designed as g-BDI agents, was presented.*

   We have proposed a software engineering process to develop graded BDI agents in a multiagent scenario. The aim of the proposed methodology is to guide the design of a multiagent system starting from a real world problem. The methodology presented has been built by adapting and extending previous approaches [88, 118, 159] in order to engineer agents with a more complex internal architecture. Our work was inspired in some sense by the design process used in [141] where the social aspects of design are considered, and the System Design phase is clearly separated from the Agent Design phase. Then, the proposed methodology is composed by two fundamental phases: the System Design phase, that has the purpose of determining the agent types composing the system and the Agent Design phase that is focussed on modelling g-BDI agents. We extract the necessary elements from the System Design phase to design the different types of agents using the proposed architecture. This process is done in two stages. The first one, deals with the logical skeleton of the multi-context specification of g-BDI model. The second one, following a flow "goals-feasible plans-beliefs-intentions" complete the agent design, filling the contents (theories) of the different contexts.

   Furthermore, the proposed process to develop g-BDI agents contributes to bridge the gap from the external functionalities assigned to a particular agent (in the System design phase), to the elements that composed the architecture (in the Detailed design stage). Besides, we have designed and implemented a case study in the tourism domain so as to show how the proposed methodology works.

Through the design and implementation of a Tourism recommender system, where one of its principal agents was modelled as a g-BDI agent (see Chapters 11 and 12), we have come all the way from the formal g-BDI model to a concrete agent implementation.

Then, the validation and experimentation of g-BDI agents was carried out by using as a case study this recommender system (detailed in Chapter 13). First, the results of the validation performed allowed us to conclude that g-BDI agents are useful to build concrete agents in real world applications. Second, we have also performed a sensitivity analysis that showed that a g-BDI agent architecture can engineer agents having different behaviors by suitably tuning some of its components. The results of a third experiment gave support to our claim that "the distinctive feature of recommender systems modelled using g-BDI agents, which is using graded

mental attitudes, allows them to provide better results than those obtained by non-graded BDI models".

14.2 Future work

This thesis has opened several possible new areas for further research. The main topics for future work are described below.

- Social aspects:

  We have presented in Chapter 8 preliminary work related to the socialization of the g-BDI agent model. With this aim we have included a social context to represent different kinds of trust and reputation in other agents. On the one hand, we have modelled in the social context the filtering of the agent information interchange. On the other hand, the social trust related to delegation was also considered.

  In this direction, an important topic for further work is to consider how to evaluate the trust in other agents, and how the agent updates this trust model along time, and after different interactions with the agent society.

- Dynamic aspects:

  To model agents that interacts in dynamic environments, it is important for the g-BDI agent to represent in relation to her beliefs, desires and intention, the notion of time. Then, we need to incorporate some elements of temporal logics to our logical framework.

  Another key problem related to the dynamic aspects of this agent model is to represent some intention reconsideration policy. It is clear that the agent should at times drop or reconsider some intentions. But reconsideration has a cost and is closely related to the problem of balancing the agent pro-active and reactive behavior, in relation to the environment dynamism.

- Revision in g-BDI Agents:

  As the agents interact in a dynamic and changing world, they may be capable to deal with inconsistencies. As our model is specified by using multi-context systems we consider important to set a general process for multi-context system revision an then, specialize it for the g-BDI agent model.

  In a Multi-context system, the theory of each context is composed of formulae coming from different provenance. Firstly, it has initial formulae. Then, the derivations from inner deductions will be added. Lastly, using bridge rules, other formulae may be introduced as the bridge rule's preconditions are satisfied. This set of formulae without any check may be inconsistent. Then, in a dynamic framework, it is needed a process to maintain the context consistency.

We belief that an argumentation based revision process for multi-context system is a promising direction of future work.

- Experiment the g-BDI model in other applications:

  We have designed and implemented a tourism recommender system where its main recommender agent, the Travel Assistant Agent, was modelled using our graded BDI approach. The experimentation on the case study proved that this agent model is useful to implement different and rich behaviours. Besides, we showed that the results obtained by recommender agents using graded attitudes improved those achieved by agents using non-graded ones.

  Further work is necessary to experiment with the g-BDI architecture to model agents in other domains. We have preliminary good results on the use of the g-BDI model in the design of an educational recommender system, to give recommendations about learning objects, taking into account the subject and the user's cultural and preference characteristics [8, 33, 44]. We belief that the model may be applied to other domains.

14.3 Related publications

The publications listed below are direct consequence of the evolution of this dissertation in the last four years:

- Casali A., Godo L. and Sierra C., Modelos BDI Graduados para Arquitecturas de Agentes. Proc. of ASAI 2004, 33JAIIO, 13 pg, Córdoba, Argentina, 2004.

  Also in *Revista Iberoamericana de Inteligencia Artificial, AEPIA*, N 26, Vol 9, pp 67-75, 2005.

- Casali A., Godo L. and Sierra C., Graded BDI Models For Agent Architectures. Leite J. and Torroni P. (Eds.) *CLIMA V, Lecture Notes in Artificial Intelligence LNAI 3487*, 126-143, Springer-Verlag, Berling Heidelberg, 2005.

  Also in *Proceedings of European Workshop of Multiagent Systems (EUMAS'04)*, Barcelona, Spain, 2004.

- Casali A., Godo L. and Sierra C., Multi-Context Specification for Graded BDI Agents. *Proceedings of the Doctoral Consortium - Fifth International Conference on Modeling and Using Context (CONTEXT-05)*, Research Report LIP 6, Paris, Francia, 2005.

- Casali A., Godo L. and Sierra C., Modeling Travel Assistant Agents: a graded BDI Approach. In Max Bramer (Ed.), *IFIP-AI, WCC. Artificial Intelligence in Theory and Practice.*, 415-424. Springer Verlag, 2006.

- Casali A., Deco C., Bender C. and Motz R., A Multiagent Approach to Educational Resources Retrieval. In *Proceedings del Workshop Inteligencia Artificial en Educación WAIFE, ASAI 2006, 35 JAIIO*,35-41. Mendoza, Argentina, 2006.

- Casali A., Godo L. and Sierra C., A Methodology to Engineering Graded BDI Agents. In *Proceedings WASI-CACIC 2006*, 12pag. Potrero de Funes, Argentina, 2006.

- Casali, A., Von Furth, A., Godo, L., Sierra, C., A Tourism Recommender Agent: From theory to prectice. In *Proceedings WASI-CACIC 2007* 1548-1561. Corrientes, Argentina, 2007.

  A revised and extended version will be published in *Revista Iberoamericana de Inteligencia Artificial, AEPIA*, to apear, 2008.

- Casali A., Godo L. and Sierra C., A Language for the Execution of Graded BDI Agents. In *Proceedings of Formal Approaches to Multi-Agent Systems FAMAS'007*, 65-82, Durham, UK, 2007.

- Casali A., Godo L. and Sierra C., A Logical Framework to Represent and Reason about Graded Preferences and Intentions. . In *Principles of Knowledge Representation and Reasoning: Proceedings of the 11th Internacional Conference (KR 2008)*, G. Brewka and J. Lang (Eds.), The AAAI Press, pp.27-37, 2008.

- Casali A., Godo L. and Sierra C., Validation and Experimentation of a Tourism Recommender Agent based on a Graded BDI Model. In: *ArtificiaL Intelligence Research and Developement*, T. Alsinet et al (Eds.), Series: Frontiers in Artificial Inteligence and Applications 184, IOS Press, pp. 41-50, 2008.

  Also in Proceedings of *XXXIV Conferencia Latinoamericana de Informtica (CLEI 2008)*, pp. 30-39, Santa Fe, Argentina, 2008.

- Deco C., Bender C., Casali A., Motz R. Design of a Recommender Educational System. Design of a Recommender Educational System. Proceedings of *3ra. Conferencia Latinoamericana de Objetos de Aprendizaje (LACLO 2008)*, Aguascalientes, Mexico, to apear, 2008.

Bibliography

[1] Acebo, E. and de la Rosa, J. L., A Fuzzy System Based Approach to Social Modeling in multiagent Systems. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS'02), Palazzo Re Enzo, Italy, 2002.

[2] Alsinet T., Chesñevar C., Godo L., Sandri S, and Simari G., Modeling Defeasible Argumentation within a Possibilistic Logic Framework with Fuzzy Unification, in *Proc. 11th Intl. Conference IPMU 2006*, 1228-1235, 2006.

[3] Amgoud L., A formal framework for handling conficting desires. In (T. D. Nielsen and N. L. Zhang, eds.), *Proceedings of ECSQARU*, volume 2711 of LNCS, 552-563. Springer, Germany, 2003.

[4] Amgoud L. and Kaci S., On the generation of bipolar goals in argumentation-based negotiation. In (I. Rahwan et al, eds.) *Proceedings of the 1st Int. Workshop on Argumentation in Multi-Agent Systems* (ArgMAS), volume 3366 of LNCS, 192-207. Springer, Germany, 2004.

[5] Amgoud L. and Prade H., A bipolar argumentation-based decision framework. In 11th *IPMU International Conference*, Paris, France, 2006.

[6] Amgoud L. and Prade H., Formalizing Practical Reasoning Under Uncertainty: An Argumentation-Based Approach. IAT 2007: 189-195, 2007.

[7] Beavers G., Automated theorem proving for Łukasiewicz logics. *Studia Logica*, Volume 52, Number 2, 183-195, 1993.

[8] Bender C., Deco C., Casali A., Motz R. and J. Guzmn. Una Plataforma Multiagente para la Búsqueda de Recursos Educacionales considerando Aspectos Culturales. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*(TEyET) Vol 1:1, 20-29, 2006.

[9] Benferhat S., Dubois D. and Prade, H., Towards a possibilistic logic handling of preferences. *Applied Intelligence* 14(3):303-317, 2001.

[10] Benferhat S., Dubois D., Kaci S. and Prade H., Bipolar Possibilistic Representations. *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence* (UAI 2002): 45-52. Morgan Kaufmann 2002.

[11] Benferhat S., Dubois D., Kaci S. and Prade H., Bipolar representation and fusion of preferences in the possilistic Logic framework. *In Proc. of the 8th International Conference on Principle of Knowledge Representation and Reasoning* (KR-2002), 421-448, 2002.

[12] Benferhat S., Dubois D., Kaci S. and Prade H., Bipolar possibility theory in preference modeling: Representation, fusion and optimal solutions. *Information Fusion*, Elsevier, 7, 135-150, 2006.

[13] Bergenti F., Gleizes M. P. and Zambonelli F. (Eds.), *Methodologies and Software Engineering For Agent Systems: The Handbook of Agent-oriented Software Engineering*, Kluwer Academic Publishing (New York, NY), July 2004.

[14] Berka T., Plobnig M. Designing Recommender Systems for Tourism. In: *ENTER 2004*, Kairo, 2004.

[15] Bhargava R. P. and Srivastava M. S., On Tukey's Confidence Intervals for the Contrasts in the Means of the Intraclass Correlation Model. *Journal of the Royal Statistical Society*. Series B (Methodological), Vol. 35:1, pp. 147-152, 1973.

[16] Bistarelli S., Pini M.S., Rossi F. and Brent Venable K., Bipolar preference problems: framework, properties and solving techniques, *LNCS* Volume 4651/2007, 78-92, Springer-Verlag, Berling Heidelberg, 2007.

[17] Booch G., Rumbaugh J. and Jacobson I., *The Unified modelling Language*. Addison-Wesley, 1999.

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

[19] Boutilier C., Towards a logic for qualitative decision theory. In *Proc. of International Conference on Principle of Knowledge Representation and Reasoning* (KR'94), 7586, 1994.

[20] Bratman, M. E., *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[21] Bratman M. E., Israel D. J. and Pollack M. E., Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4: 349-355, 1988.

[22] Bratman M. E. What is intention? In *Intention in Communication* (P.R. Cohen, J.L. Morgan and M.E. Pollak Eds.), pp. 15-32. MIT University Press, Cambridge, MA, 1990.

[23] Brooks R. A., A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, 2(1):14-23, 1986.

[24] Burke R., Knowledge-based recommender systems. In (A. Kent, ed.), *Encyclopedia of Library and Information Systems*, Vol. 69:32, 2000.

[25] Burke R., Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331370, 2002.

[26] Cacioppo J. T., Gardner W. L. and Bernston G. G., Beyond bipolar conceptualizations and measures: The case of attitudes and evaluative space, *Personality and Social Psychology Review* 1 (1), 325, 1997.

[27] Cacioppo J. T. and Bernston G. G., The affect system: Architecture and operating characteristics, *Current Directions in Psychological Science* 8 (5) 133137, 1999.

[28] Cardelli L. and Gordon A. D., Mobile Ambients. In (Maurice Nivat, ed.) *Foundations of Software Science and Computational Structures*, number 1378 in LNCS, 140155. Springer-Verlag, 1998.

[29] Casali A., Godo L. and Sierra C., Modelos BDI Graduados para Arquitecturas de Agentes. In *Proc. of ASAI 2004, 33 JAIIO* 13 pg,, Córdoba, Argentina, 2004. Also in *Revista Iberoamericana de Inteligencia Artificial, AEPIA*, N 26, Vol. 9, 67-75, 2005.

[30] Casali A., Godo L. and Sierra C., Graded BDI Models For Agent Architectures. In (Leite J. and Torroni P., Eds.), *CLIMA V, Lecture Notes in Artificial Intelligence* LNAI 3487, 126-143, Springer-Verlag, Berling Heidelberg, 2005. Also in *Proceedings of European Workshop of Multiagent Systems* (EUMAS'04), Barcelona, Spain, 2004.

[31] Casali A., Godo L. and Sierra C., Multi-Context Specification for Graded BDI Agents. In *Proceedings of the Doctoral Consortium - Fifth International Conference on Modeling and Using Context* (CONTEXT-05) Research Report LIP 6, Paris, Francia, 2005.

[32] Casali A., Godo L. and Sierra C., Modeling Travel Assistant Agents: a graded BDI Approach. In Max Bramer (Ed.), *IFIP-AI, WCC. Artificial Intelligence in Theory and Practice.*, 415-424. Springer Verlag, 2006.

[33] Casali A., Deco C., Bender C. and Motz R., A Multiagent Approach to Educational Resources Retrieval. In *Proceedings del Workshop Inteligencia Artificial en Educación WAIFE, ASAI 2006, 35 JAIIO*, 35-41. Mendoza, Argentina, 2006.

[34] Casali A., Godo L. and Sierra C., A Methodology to Engineering Graded BDI Agents. In *Proceedings WASI-CACIC 2006*, 12pg. Potrero de Funes, Argentina, 2006.

[35] Casali, A., Von Furth, A., Godo, L., Sierra, C., A Tourism Recommender Agent: From theory to prectice. In *Proceedings WASI-CACIC 2007* 1548-1561. Corrientes, Argentina, 2007. Also in *Revista Iberoamericana de Inteligencia Artificia, AEPIA* to apear, 2008.

[36] Casali A., Godo L. and Sierra C., A Language for the Execution of Graded BDI Agents. In *Proceedings de Formal Approaches to Multi-Agent Systems FAMAS'007*, 65-82, Durham, UK, 2007.

[37] Casali A., Godo L. and Sierra C., A Logical Framework to Represent and Reason about Graded Preferences and Intentions. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning, KR 2008*, Sydney, Australia, to apear, 2008.

[38] Casali A., Godo L. and Sierra C., Validation and Experimentation of a Tourism Recommender Agent based on a Graded BDI Model. In *Proceedings of 11th International Conference of the Catalan Association for Artificial Intelligence (CCIA 2008)*, Sant Mart d'Empries, Espaa, to apear, 2008.

[39] Castelfranchi C. and Falcone R., Social Trust: A Cognitive Approach, in *Trust and Deception in Virtual Societies*, Kluwer Academic Publishers, 55-90, 2001.

[40] Cimatti A. and Serafini L., Multi-Agent Reasoning with Belief Contexts: the Approach and a Case Study. In (M. Wooldridge and N. R. Jennings, eds.) *Intelligent Agents: Proceedings of 1994 Workshop on Agent Theories, Architectures, and Languages*, in LNCS number 890 , 71-5. Springer Verlag, 1995.

[41] Cohen, P. R. and Levesque, H. J., Intention is choice with commitment. *Artificial Intelligence*, 42:213-261, 1990.

[42] Cohen, P. R. and Levesque, H. J., Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 65-72, Menlo Park, CA, AAAI Press, 1995.

[43] Dastani M., Herzig A., Hulstijn J. and van der Torre L., Inferring Trust. In Leite J. and Torroni P. (Eds.), *Proceedings of the 5th International Workshop on Computational Logic in Multiagent Systems (CLIMA V)*, 34-49, 2004.

[44] Deco C., Bender C., Casali A., Motz R., Design of a Recommender Educational System. In *Proceedings of 3ra. Conferencia Latinoamericana de Objetos de Aprendizaje (LACLO 2008)*, Aguascalientes, Mexico, to apear, 2008.

[45] Dennet D. C., *The Intentional Stance*. MIT Press, Cambridge, MA, 1987.

[46] d'Inverno M. et al., A Formal Specification of dMARS. In Rao A., Singh P. and Wooldridge M. (Eds.), *Intelligent Agents, IV* , LNAI Volume 1365, 155-176, Springer, Berlin, 1997.

[47] Dellunde P. and Godo L., Introducing Grades in Deontic Logics. In R. van der Meyden and L. van der Torre (Eds.), *Proc. of the Ninth International Conference on Deontic Logic in Computer Science (DEON'08)*, LNAI, Springer, to appear, 2008.

[48] Dennet, D. C., *The Intentional Stance*. MIT Press, Cambridge, MA, 1987.

[49] Doyle J., Shoham Y. and Wellman., A Logic os Relative Desire. In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, 16-31, 1991.

[50] Dubois, D. and Prade, H., Possibility theory: Qualitative and quantitative aspects. In Smets, Ph. (Ed.), *Quantified Representations of Uncertainty and Imprecision*, vol. 1, 169-226. Kluwer, Dordrecht, 1998.

[51] Dubois D. and Fargier H., Qualitative Decision Making with Bipolar Information. In *KR 2006*, 175-186, 2006.

[52] Dung P. M., On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321-358, 1995.

[53] Emerson E. A. and Strinivasan J., Branching time temporal logic. In J. W. de Bakker, W. P. de Roever and G. Rozenberg (Eds.), *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, 123-172. Springer -Verlag, 1989.

[54] Esteva, F., Garcia, P. and Godo L., Relating and extending semantical approaches to possibilistic reasoning. *International Journal of Approximate Reasoning*, 10:311-344, 1994.

[55] Esteva M., Padget J. and Sierra C., Formalizing a Language for Institutions and Norms. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures and Languages (ATAL-2001)*, 106-119, 2001.

[56] Falappa M., Kern-Isberner G. and Simari G., Belief revision, explanations and defeasible reasoning. *Artificial Intelligence Journal*, 141:128, 2002.

[57] Felfernig A., Gordea S., Jannach D., Teppan E. and Zanker M., A short Survey of Recommendation Technologies in Travel and Tourism, in *Oesterreichische Gesellschaft fuer Artificial Intelligence OEGAI Journal*, Vol. 25, No. 7, 17-22, 2007.

[58] Felfernig A., Friedrich G., Jannach D. and Zanker M., An Environment for the Development of Knowledgebased Recommender Applications, to appear in *International Journal of Electronic Commerce (IJEC)*, 2007.

[59] Ferguson I. A., TouringMachines: an Architectures for Dynamic, Rational , Mobile Agents. *PhD Thesis, Clare Hall, University of Cambrage, UK.* (Also available as Technical Report n 273, University of Cambrage Computer Laboratory).

[60] García A. and Simari G., Defeasible logic programming: An argumentative approach. *Theory Practice of Logic Programming* 4(1):95138, 2004.

[61] Georgeff M. P. and Lansky A. L., Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 677682, Seattle, WA, 1987.

[62] Georgeff M. P. and Rao A. S., A profile of the Australian AI Institute. *IEEE Expert*, 11(6):8992, December 1996.

[63] Halpern J. Y., *Reasoning about Uncertainty*. The MIT Press. Cambridge Massachusetts, 2003.

[64] Harel D., Dynamic logic, in Gabbay D. and Guenthner F. (Eds.), *Handbook of Philosophical Logic* Vol. II 497-604, 1984.

[65] Howden N., Rönnquist R., Hodgson A. and Lucas A., JACK Summary of an Agent Infrastructure, in *Proceedings of the 5th International Conference on Autonomous Agents*, 2001.

[66] Georgeff M., Pell B., Pollack M., Tambe M. and Wooldridge M., The Belief-Desire-Intention Model of Agency. In J. P. Muller, M. Singh, and A. Rao (Eds.), *Intelligent Agents V - LNAI*, Volume 1365, Springer-Verlag, 1999.

[67] Gerla B., Rational Łukasiewicz logic and DMV-algebras. *Neural Networks World*, vol. 11:579-584, 2001.

[68] Ghidini C. and Giunchiglia F., Local Model Semantics, or Contextual Reasoning = Locality + Compatibility *Artificial Intelligence*,127(2):221-259, 2001.

[69] Giovannucci A. Towards Multi-Context based Agents Implementation. *IIIA-CSIC Research Report*, 2005.

[70] Giunchiglia F., Contextual Reasoning. In *Proceedings of the IJCAI Workshop on Using Knowledge in Context*, 1993.

[71] Giunchiglia F. and Serafini L., Multilanguage Hierarchical Logics (or: How we can do without modal logics) *Journal of Artificial Intelligence*, vol.65, pp. 29-70, 1994.

[72] Godo L., Esteva F. and Hajek P., Reasoning about probabilities using fuzzy logic. *Neural Network World*, 10:811-824, 2000.

[73] Godo L., Esteva F. and Hajek P., A Fuzzy Modal Logic for Belief Functions, *Fundamenta Informaticae archive*, Vol. 57: 2-4, 127-146, 2003.

[74] Goldblatt R., *Logics of Time and Computation*, CSLI Lecture Notes 7, 1992.

[75] Gottwald, S., *A Treatise on Many-valued Logics*. Studies in Logic and Computation. Research Studies Press, Baldock, 2001.

[76] Griffiths N. and Luck M., Cooperative Plan Selection Through Trust. In *Proceedings of MAAMAW-99*, 162-174, Spain, 1999.

[77] Hájek P., Godo L. and Esteva F., Fuzzy logic and probabitity, In *Proc. of the International Conference on Uncertainty in Artificial Intelligence (UAI 95)*, 237-244, Montreal, Canada, 1995.

[78] Hájek P., Metamatematics of Fuzzy Logic, *Trends in Logic*, 4, Kluwer Academic Publishers (1998).

[79] Halpern J. Y., *Reasoning about Uncertainty*. The MIT Press. Cambridge Massachusetts, 2003.

[80] Harel D., Dynamic logic, in Gabbay D. and Guenthner F. (Eds.), *Handbook of Philosophical Logic* Vol II 497-604, 1984.

[81] Huber M. J., JAM: A BDI-theoretic Mobile Agent Architecture *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, 236-243. Seattle, WA, 1999.

[82] Hulstijn, J., and van der Torre, L. 2004. Combining goal generation and Planning in and argumentation framework. In A. Hunter and J. Lang (Eds.), *Proc. Workshop on Argument, Dialogue and Decision, NMR*, Whistler, Canada, 2004.

[83] Iglesias C., Garijo M., and Gonzalez J., A Survey of Agent-Oriented Methodologies. In Intelligent Agents V - *Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, LNAI. Springer-Verlag, Heidelberg, 1999.

[84] Ingrand F. F., Georgeff M. P. and Rao A. S., An architecture for real time reasoning and system control. *IEEE Expert*, 7(6), 1992.

[85] Jennings N.R., On Agent-Based Software Engineering. *Artificial Intelligence* 117(2), 277-296, 2000.

[86] Jennings N. R., Sycara K. and Wooldridge M., A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*. 1(1), 7-38, 1998.

[87] Jo Ch. H., A Seamless Approach to the Agent Development, *ACM Symposium on Applied Computing SAC 2001*, Las Vegas, 641-647, 2001.

[88] Jo Ch. H., Chen G. and Choi J., A New Approach to the BDI Agent-BAsed modelling. *ACM Symposium on Applied Computing SAC'04*, ACM 1-58113-812-1/03/04, 1541-1545 Nicosia, Cyprus, 2004.

[89] Jo, Ch. H. and Einhorn, J. M., A BDI Agent-Based Software Process, in *Journal of Object Technology*, vol. 4, no. 9, 2005.

[90] Joseph S., Perreau de Pinninck A., Robertson D., Sierra C., Walton C., Interaction Model Language Definition. In Dignum V., Dignum F., Matson E. and Edmonds B. (Eds.), *IJCAI 2007 Workshop AOMS Agent Organizations Models and Simulations*, 49-61, 2007.

[91] Kakas A. and Moraitis P., Argumentation based decision making for autonomous agents. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 883-890, Melbourne, Australia, 2003.

[92] Kendall E. A., Agent Software Engineering with Role modelling. In Ciancarini P. and Wooldridge M. (Eds.), *Proceedings of Agent Oriented Software Engineering (AOSE-2000)*, 163-169. Springer-Verlag, Berlin, Germany, 2000.

[93] Kinny D., Georgeff M., and Rao A., A Methodology and modelling Techniques for Systems of BDI agents. *Proc. of the 7th European Workshop on modelling Autonomous Agents in a Multi-Agent World*, LNAI Vol. 1038: 56-71, Springer, 1996.

[94] Knijnenburg P. M. W. and van Leenwen J., On models for Propositional Dynamic Logic, *Theor. Comput. Sci.*, 91(2)181-203, 1991.

[95] Lang J., Conditional Desires and Utilities - an alternative logical approach to qualitative decision theory. In *Proceedings of 12th European Conference on Artificial Intelligence (ECAI'96)*, 318-322, 1996.

[96] Lang J., van der Torre, L. and Weydert E., Utilitarian Desires, *Autonomous Agents and Multi Agent systems*, vol. 5:3, 329-363, 2002.

[97] Lang J., van der Torre, L. and Weydert E., Hidden Uncertainty in the Logical Representation of Desires *International Joint Conference on Artificial Intelligence (IJCAI 03)*, Acapulco, Mexico, 2003.

[98] Liau C. J., Belief, Information Acquisition, and Trust in Multiagent Systems - a modal formulation. *Artificial Intelligence*, 149, 31-60, 2003.

[99] Ljungberg M. and Lucas A., The OASIS Air-traffic Management System. In *Proceedings of the 2nd Pacific Rim International Conference on Artificial Intelligence(PRICAI92)* , Scoul, Korea, 1992.

[100] López de Mántaras R., *Approximate Reasoning Models*. Ellis Horwood, 1990.

[101] Maes P., The Agent netword architecture (ANA). *SIGART Bulletin*, 2(4) 115-120, 1991.

[102] Luck M., McBurney P. and Preist C., Agent Technology: Enabling Next Generation Computing. *AgentLink, 2003*, ISBN 0854 327886. (http://www.agentlink.org/roadmap/)

[103] Mazlack L.J., Lee W. and Pick R.A., Recognizing the most effective approximate reasoning calculi for a knowledge-based system. *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, 1991.* Volume 3: 8-11, 349-358, 1991.

[104] Meyer J. J., Dynamic Logic for Reasoning about Actions and Agents. *Workshop on Logic-Based Artificial Intelligence*, Washington, DC, 1999.

[105] Mladenic D., Text-learning and related intelligent agents. IEEE Intelligent Systems, 14:4454, 1999.

[106] Milner R., Parrow J. and Walker D., A calculus of mobile processes, Parts 1-2. *Information and Computation*, 100(1), 1-77. 1992.

[107] Milner R., *Communication and Comcurrency*. Prentice-Hall International, 1989.

[108] Montaner M., López B. and de la Rosa J. L., A Taxonomy of Recommender Agents on the Internet, *Artificial Intelligence Review*, Kluwer Academic Publishers. Vol. 19:4, 285-330, 2003.

[109] Montaner, M., Collaborative Recommender Agents Based On Case-Based Reasoning and Trust. *PhD Thesis in Computer Engineering. Departament of Electronics, Computer Science and Automatic Control. Universitat de Girona.* September, 2003.

[110] Mooney R. J. and Roy L., Content-based book recommending using learning for text categorization, in *Proceedings of the fifth ACM conference on Digital libraries Publisher*, ACM Press, 195-204, 2000.

[111] Muller J.P., Pischel, M. and Thiel M., Modeling Reactive Behaviour in Vertically Layered Agent Architectures. In Wooldridge M. and Jennings N. (Eds.), *Intelligent Agents: theories, Architectures and Languages*, LNAI Volume 890, 261-276. Springer, Berlin, 1995.

[112] Muller J.P., A cooperation Model for Autonomous Agent. In J. P. Muller, M. Wooldridgde and N. R. Jennings (Eds.), *Intelligent Agents, III*, LNAI Volume 1193, 245-260, Springer, Berlin, 1997.

[113] Niinivaara O., Agent-Based Recommender Systems. *Technical Report, University of Helsinki*, Dept. of CS., 2004.

[114] Noriega P. and Sierra C., Towards layered dialogical agents. In Muller J. P., Wooldridge M. J., and Jennings N. R. (Eds.) *Intelligent Agents III* , 173-188, Springer Verlag, Berlin, 1996.

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

[116] Parsons S. and Giorgini P., On using degrees of belief in BDI agents. *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, 1998.

[117] Parsons S., Pettersson O., Saffiotti A. and Wooldridge M., Intention Reconsideration in Theory and Practice. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, IOS Press : 378-382, Berlin, Germany, 2000.

[118] Padgham L. and Winikoff M., Prometheus: A Methodology for Developing Intelligent Agents. Proceedings of the first international joint conference on Autonomous agents and multiagent systems, 2002.

[119] Parsons S., Jennings N.J., Sabater J. and Sierra C. Agent Specification Using Multi-context Systems. *Foundations and Applications of Multi-Agent Systems 2002*, 205-226, 2002.

[120] Pinyol I. and Sabater-Mir J., Integrating Image and Reputation Information in BDI Agents *EUMAS 08*, Bath,UK, to apear, 2008.

[121] Prantner K., E-Tourism Working Draft, DERI Innsbruck (www.deri.org), 2004.

[122] Rahwan, I., and Amgoud, L., An Argumentation based Approach for Practical Reasoning. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, 347-354, 2006.

[123] Rao A. and Georgeff M., Modeling Rational Agents within a BDI-Architecture. In R. Fikes and E. Sandewall (Eds.), *proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, 473-484 , Morgan Kaufmann, San Mateo, CA, 1991.

[124] Rao, A. and Georgeff M., Deliberation and Intentions. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, 1991.

[125] Rao, A. and Georgeff M., BDI agents: From theory to practice. In *proceedings of the 1st International Conference on Multi-Agents Systems*, pp 312-319, 1995.

[126] Rao A. S. and Georgeff M., Decision procedures of BDI logics. *Journal of Logic and Computation*, 8(3):293344, 1998.

[127] Resnick P. and Varian H., Recommender Systems. In *Communications of the ACM*, 56-58, 1997.

[128] Ricci F., Travel recommender Systems, in *IEEE Intelligent Systems*, November/December 2002, 55-57, 2002.

[129] Ricci F., and Del Missier F., Supporting Travel Decision Making through Personalized Recommendation. In C. M. Karat, J. Blom, and J. Karat (Eds.), *Designing Personalized User Experiences for eCommerce*, Kluwer Academic Publisher, 221-251, 2004.

[130] Robertson D., Multi-Agent Coordination as Distributed Logic Programming, in *Proceedings of the International Conference on Logic Programming*, Sant-Malo, 2004.

[131] Rotstein N., and García A., Defeasible reasoning about beliefs and desires. In *Proc. of 11th NMR*, 429436, 2006.

[132] Rotstein N. D., García A. J. and Simari G. R., Reasoning from Desires to Intentions: A Dialectical Framework. *Twenty-Second AAAI Conference on Artificial Intelligence* (AAAI-07). Vancouver, BC, Canada. Pp. 136141. 2007.

[133] Russell, S. and Norvig P., *Artificial Intelligence: a modern approach*. Prentice Hall. Englewoods Cliifs, NJ, 1995.

[134] Simari G. R., García A. J., and Capobianco M., Actions, planning and defeasible reasoning. In *Proc. 10th International Workshop on Non-Monotonic Reasoning*, 377-384, 2004.

[135] Sabater J. and Sierra C., Regret: A reputation model for gregarious societies. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, 61-69, Montreal, Canada, 2001.

[136] Sabater J., Sierra C., Parsons S. and Jennings N. R., Engineering executable agents using multi-context systems. *Journal of Logic and Computation* 12(3): 413-442, 2002.

[137] Sabater J. and Sierra C., Reputation and social network analysis in multiagent systems. *Proceedings AAMAS'02*, 475-482, Bologna, Italy, 2002.

[138] Sabater J., Trust and Reputation for Agent Socities. *PhD Thesis Sabater J., Artificial Intelligence Institute Monograph*, IIIA-CSIC N20, 2003.

[139] Schild K., On the relationship between BDI logics and standard logics of concurrency. In J. P. Muller, M. P. Singh, and A. S. Rao (Eds.), *Intelligent Agents V Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, LNAI Springer-Verlag, Heidelberg, 1999.

[140] Sierra C., Godo L., López de Mántaras R. and Manzano, Descriptive Dynamic Logic and its Application to Reflective Architectures. *Future Generation Computer Systems*, 12, 157-171, 1996.

[141] Sierra C., Thangarajah J., Padgham L. and Winikoff M., Designing Institutional Multi-Agent Systems. *AOSE 2006*: 84-103, 2006.

[142] Schut, M., Wooldridge, M. and Parsons S., Reasoning About Intentions in Uncertain Domains Symbolic and Quantitative Approaches to Reasoning with Uncertainty. *Proceedings of the 6th ECSQARU 2001*, 84-95, Toulouse, France, 2001.

[143] Schut M. and Wooldridge M., Principles of intention reconsideration. In *Proc. of the 5th International Conference on Autonomous Agents*,340-347, 2001.

[144] Tang Y. and Parsons S., Argumentation-based dialogues for deliberation. In F. Dignum et al. (Eds.), *Proc. AAMAS*, Utrecht, The Netherlands, 552-559. ACM Press, New York NY, USA, 2005.

[145] Terveen L. G. and Hill W., Beyond Recommender Systems: Helping People Help Each Other. In Carroll, J. (Ed.), *HCI in the New Millennium*. Addison Wesley, 2001.

[146] van der Torre L. and Weydert E., Parameters for Utilitarian Desires in a Qualitative Decision Theory. *Applied Intelligence*, 14:285-301, 2001.

[147] van Linder B., Modal Logics for Rational Agents, *PhD. Thesis*, Utrech University 1996.

[148] Walton C. and Robertson D., Flexible multi-agent protocols. *Technical Report EDI-INF-RR-0164*, University of Edinburgh, 2002.

[149] Walton C., Multi-Agent Dialogue Protocols. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, January 2004.

[150] Walton C., Model Checking Multi-Agent Web Services. In *Proceedings AAAI Spring Symposium on Semantic Web Services*, Stanford, California, 2004.

[151] Walton C., Protocols for Web Service Invocation, in *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web (ASW05)*,6 pag., Arlington, USA, November 2005.

[152] Weiss G., In Weiss G. (Ed.), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 1999.

[153] Werthner H., Intelligent Systems in Travel and Tourism, in *Proceeding of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03*, 1620-, Acapulco, Mexico, 2003.

[154] Wooldridge M and Jennings N. R., Intelligent Agents: theory and practice. *The Knowledge Engineering Review*, 10(2), 115-152, 1995.

[155] Wooldridge M. and Jennings N. R., Agent-based software engineering. *IEEE Proceedings in Software Engineering*, 144(1), 26-37, 1997.

[156] Wooldridge, M. and Parsons S., Intention Reconsideration Reconsidered. In *Proceedings of Intelligent Agents V, 5th International Workshop Agent Theories, Architectures, and Languages (ATAL '98)*, Paris, France, 1998. Jrg P. Mller, Munindar P. Singh, Anand S. Rao (Eds.), LNCS 1555, 63-79, Springer, 1999.

[157] Wooldridge M. J., Jennings N. R. and Kinny D., The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems (Agents'99)*, Vol. 3 (3) 285 - 312, Kluwer, 2000.

[158] Wooldridge, M., *Introduction to Multiagent Systems*, John Wiley and Sons,Ltd., 2001

[159] Zhang T., Kendall E. and Jiang H., A Software Engineering Process for BDI Agent-Based Systems, in *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology(IAT'03)*, 0-7695-1931-8/03IEEE, 2003.