A Language for the Execution of Graded BDI Agents

ANA CASALI, *FCEIA and CIFASIS, Universidad Nacional de Rosario (UNR), Av Pellegrini 250, 2000 Rosario, Argentine.*
E-mail: acasali@fceia.unr.edu.ar

LLUÍS GODO, *IIIA - CSIC, Campus de la UAB s/n, 08193 Bellaterra, Spain. E-mail: godo@iiia.csic.es*

CARLES SIERRA, *IIIA - CSIC, Campus de la UAB s/n, 08193 Bellaterra, Spain. E-mail: sierra@iiia.csic.es*

Abstract

We are interested in the specification and deployment of multi-agent systems, and particularly we focus on the execution of agents. Along this research line we have proposed a general model for *graded BDI agents*, specifying an architecture based on multi-context systems (MCS) and able to deal with the environment uncertainty (via graded beliefs) and with graded mental proactive attitudes (via desires and intentions). These graded attitudes are represented using appropriate fuzzy modal logics. In this paper we cope with the operational semantics of this agent model. We present a Multi-context Calculus, based on Ambient Calculus, for the execution of multi-context systems with its corresponding semantics. This calculus is general enough to support different kinds of MCSs and particularly, we show how a graded BDI agent can be mapped into the language of the calculus.

Keywords: Operational semantics, Multi-context systems, BDI agents, Ambient calculus

1 Introduction

In order to achieve the full potential of agent approaches and technologies many different theories, architectures and infrastructures are required to specify, design and implement agent-based systems. In this context we focus on the execution of intentional agents. We consider that making the BDI architecture [28] more flexible will allow us to design and develop agents potentially capable to have a better performance in uncertain and dynamic environments. Along this research line we have proposed a general model for *graded BDI agents* (g-BDI agents for short), specifying an architecture able to deal with the environment uncertainty (via graded beliefs) and with graded mental proactive attitudes (via desires and intentions). In the g-BDI model, Belief degrees represent the extent to which the agent believes formulas hold 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)

[18], 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. The logical framework of this model has been presented in [8, 12] and how this model can be used to specify an architecture for a Travel Assistant Agent can be seen in [10]. This agent architecture can be extended to include the social aspects of agency by adding a social context to represent different kinds of trust in other agents [9, 10]. Further research in this direction is ongoing and a relevant work to be applied to represent a social trust theory is the intentional agent model proposed by Lorini et al. in [22]. Also, we consider very promising the work of Pinyol and Sabater-Mir in [27] focusing on the integration of a cognitive model of reputation within a BDI agent architecture. Moving to the multi-agent systems framework, a very relevant formal approach to deal with teams of agents with different types of group beliefs and collective Intentions is developed in [14].

In this work we cope with the operational semantics aspects of the g-BDI agent model. As this model is based on multi-context systems [26] and these systems are basically deductive machines, we want to introduce an specification language that allows us to define the computational meaning of an agent. The semantics for a g-BDI model of agent will describe how a valid agent is interpreted as sequences of computational steps. These sequences are the meaning of the model. Operational semantics define an abstract machine and give meaning to language expressions by describing the transitions they induce on states of the machine. Alternatively, with different process calculi, semantics are defined via syntactic transformations on expressions of the language itself.

The process calculus approach has been used to cope with formal aspects of multiagent interactions. For example, we can mention some relevant approaches. The π-calculus developed by Milner et al. [23] is able to describe concurrent computations whose configuration may change during the computation. The Ambient Calculus due to Cardelli et al. [6] describes the movement of processes (agents) and devices. The Lightweight Coordination Calculus (LCC) [29] allows an asynchronous semantics to coordinate processes that may individually be in different environments. Walton in [36] presents a language based on CCS [24] to specify agent protocols in a flexible manner during the interaction of agents. Ambient LCC [21] was specially designed to support the execution of electronic institutions. To give semantics to our g-BDI agent model we decided to follow this process algebra approach.

Since the g-BDI agent model is formalized using multi-context systems, 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 multi-context systems and particularly, we show how a graded BDI agent can be mapped to it.

This paper is organized as follows. In Section 2 we overview the g-BDI model of agent based in multi-context systems. Section 3 outlines some process calculus related to multiagent systems. In Section 4 we present the Multi-context Calculus, and in Section 5 we provide a semantics for this calculus. The mapping from g-BDI agents to Multi-Context Calculus is presented in Section 6 and, finally, in Section 7 some conclusions are outlined.

2 Graded BDI agent model

The graded BDI model of agent (g-BDI) allows to specify agent architectures able to deal with the environment uncertainty and with graded mental attitudes. In this sense, belief degrees represent to what extent the agent believes a formula is true. Degrees of positive or negative desire allow the agent to set different levels of preference or rejection respectively. Intention degrees give also a preference measure but, in this case, modeling the cost/benefit trade off of reaching an agent's goal. Thus, a higher intention degree towards a goal means that the benefit of reaching it is high, or the cost is low. Then, Agents having different kinds of behaviour can be modeled on the basis of the representation and interaction of these three attitudes.

The specification of the g-BDI agent model is based on multi-context systems (MCS) since we consider that the modelling of different intentional notions by means of several modalities (B, D, I) can be very complex if only one logical framework is used. Multi-context systems were introduced by Giunchiglia and Serafini [18] to allow different formal (logic) components to be defined and interrelated. MCS are close to Gabbay's Labelled Deduction Systems (LDS) [15] in the sense of providing a framework for presenting and using different logics in a uniform and natural way as LDS do. MCS have been used in different applications, as for example in the integration of heterogeneous knowledge and data bases, in default reasoning [4], in the formalization of reasoning about beliefs (more generally, propositional attitudes) [16], or for engineering agents in multiagent systems [26].

The MCS specification contains two basic components: units (or contexts) and bridge rules, which channel the propagation of consequences among unit theories. Thus, a MCS is defined as a group of interconnected units $\langle \{C_i\}_{i\in I}, \Delta_{br} \rangle$. Each context C_i is specified by a 3-tuple $C_i = \langle L_i, A_i, \Delta_i \rangle$ where L_i, A_i and Δ_i are its language, axioms, and inference rules respectively. Δ_{br} can be understood as rules of inference with premises and conclusions in different contexts, for instance a bridge rule like

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

specifies 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 MCS is complete.

The deduction mechanism of a multi-context system $\langle \{C_i\}_{i\in I}, \Delta_{br} \rangle$ is therefore based on two kinds of inference rules, internal rules Δ_i, and bridge rules Δ_{br}, which allow to embed formulae into a context whenever the conditions of the bridge rule are satisfied.

In the basic specification of a g-BDI agent model as a MCS we have two kinds of contexts: three *mental* contexts, to represent beliefs (BC), desires (DC) and intentions (IC), as well as two *functional* contexts, for planning (PC) and communication (CC). The overall behavior of the system will depend of the logical representation of each intentional notion in their corresponding contexts and the particular set of bridge rules Δ_{br} used. Thus, a g-BDI agent model will be defined as a MCS of the form

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

Figure 1 illustrates such a g-BDI agent model with the different five contexts and six bridge rules relating them.

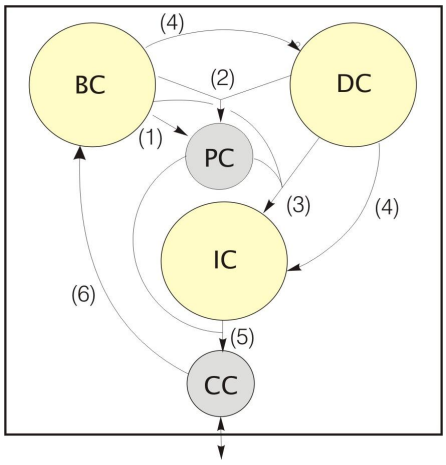

FIG. 1. Multi-context model of a graded BDI agent

Next, we synthesize the purpose of each component (i.e. contexts and bridge rules) in the agent model.

Belief Context (BC): the aim of this context is to model the agent's uncertain beliefs about the environment. Since the agent needs to reason about her possible actions and the environment transformations they cause and their associated cost, this knowledge must be part of any situated agent's belief set.

Desire Context (DC): in this context, we represent the agent's desires. Desires represent the agent's *ideal* preferences regardless of the agent's current perception of the environment and regardless of the cost involved in actually achieving them. Following a bipolar representation of preferences [3], we formalize agent's 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. Both, positive and negative desires can be graded.

Intention Context (IC): this unit is used to represent the agent's intentions. Together with the desires, they represent the agent's preferences. However, we consider that intentions cannot depend just on the benefit of reaching a goal φ, but also on the world's state and the cost of transforming it into one 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 towards the goal. A suitable bridge rule (described bellow as Bridge rule (3)) infers these degrees of intention towards a goal φ for each plan α that allows to achieve the goal.

Planner and Communication Contexts (CC and PC): the Planner Context has to look for feasible plans, these plans are generated from actions that are believed to fulfill a given positive desire and avoiding negative desires as post-conditions. These feasible plans are computed within this unit using an appropriate planner

that takes into account beliefs and desires injected by bridge rules from the BC and DC units respectively.

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. The theory inside this context will take care of the sending and receiving of messages to and from other agents in the multiagent society where our graded BDI agent lives.

Bridge rules (BRs): we define a collection of basic bridge rules to establish the necessary interrelations between context theories. As for example we describe one of them (see Bridge rule (3) in Figure 1):

- 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 satisfying the (positive) desire φ, the cost of the plan and the belief degree in the full achievement of φ after performing α. The bridge rule (3) in Fig. 2 computes this value from the degree d of $D^+\varphi$, the degree r of the belief formula $B[\alpha]\varphi$ and the cost C of the plan α:

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

  Different functions f allow to model different agent behaviours.

In order to represent and reason about graded notions of belief, desire and intention, we use a modal many-valued approach developed by Hájek and colleagues [19, 17] where uncertainty reasoning is dealt with by defining suitable modal theories over suitable many-valued logics[1].

For instance, let us consider a Belief context where belief degrees are to be modeled as probabilities. Then, for each classical formula φ, we consider a modal formula $B\varphi$ which is interpreted as "φ is probable". This modal formula $B\varphi$ is then 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 suitable $[0, 1]$-valued logic, we can express the governing axioms of probability theory as logical axioms involving modal formulae. For instance, the following axiom over Łukasiewicz logic

$$B(\varphi \vee \psi) \equiv B\varphi \oplus (B\psi \ominus B(\varphi \wedge \psi))$$

properly captures the finite additivity property of the belief operator B (see the defintion of the Lukasiewicz logic connectives $\equiv$, $\oplus$ and $\ominus$, and their semantics in Annex I). Then, the many-valued logic machinery can be used to reason about the modal formulae $B\varphi$, which faithfully respects the uncertainty model (e.g. probabilistic, possibilistic) chosen to represent the degrees of belief.

In this proposal, for the belief context we choose the Rational Pavelka Logic (RPL) an extension of the infinitely-valued Łukasiewicz logic (see Annex I for RPL definition) as the base many-valued logic [8]. This selection is done because we adopt a

[1] Other approaches could have also been used, e.g. those based on graded (classical) probabilistic modalities [31, 25] or plausibility modalities [32], or the uncertainty logics à la Halpern [20], but the fuzzy modal approach seems to offer a uniform, flexible enough and rather elegant framework for our purposes.

probabilistic semantics of graded beliefs, but another choice could be made according to the kind of measure chosen in a particular case (see e.g. [5, 7] for more details). Indeed, to set up an adequate axiomatization for our belief context logic we need to combine axioms for the classical (crisp) formulae, axioms of Łukasiewicz logic for modal formulae, and additional axioms for B-modal formulae according to the probabilistic semantics of the B operator. The same many-valued modal logic approach is used to represent and reason under graded attitudes in the other mental contexts. The formalization of the adequate logics –language, semantics, axiomatization and rules – for the different contexts is described in [8, 12] and a synthesized description of the Desire context can be seen in next subsection 6.1.

Below we present a simple example as to show how the proposed agent model works.

Example 2.1
Peter, who lives in Rosario, wants to plan his activities for the next week. He activates a personal assistant agent based on our *g-BDI model* to find an adequate travel plan (transportation + accommodation). He would be very happy attending to a conference on his speciality scheduled to take place in Buenos Aires (φ_1) and he would be rather happy visiting a friend living near this city (φ_2). But he would indeed prefer much more to be able to do both things. Besides, he doesn't like to travel during the night (ψ). This assistant has Peter's positive and negative desires represented by the following formulae in the theory T_{DC} of the *agent*'s Desire context:

$$T_{DC} = \{(D^+\varphi_1, 0.8), (D^+\varphi_2, 0.6), (D^+(\varphi_1 \wedge \varphi_2), 0.9), (D^-\psi, 0.7)\}$$

This means that the agent has rather high positive desires on achieving φ_1 and φ_2 but he even has a higher desire to achieve both (0.9). At the same time, the agent he rather rejects ψ, represents by a rather negative desire on ψ (0.7).

The agent also has knowledge about the conference schedule, his friend's agenda and transportation information, that is represented in the theory T_{BC} of the Belief context BC. Moreover, from this information and the set of positive and negative desires in T_{DC}, the planner context (PC) looks for *feasible* travel plans that are believed to satisfy φ_1 and/or φ_2 by their execution, but avoiding ψ as post-condition. Assume both α and β are found as feasible plans, whose normalized costs are $c_\alpha = 0.6$ and $c_\beta = 0.5$ respectively.

On the other hand, assume the Belief context (BC) is able to estimate the following beliefs (modelled as probabilities) about the achievement of the different goals by the feasible plans α and β, represented by the following formulae in the theory T_{BC}:

$$\begin{aligned} T_{BC} \supseteq \{&(B[\alpha]\varphi_1, 0.7),\ (B[\alpha]\varphi_2, 0.6),\ (B[\alpha](\varphi_1 \wedge \varphi_2), 0.42),\\ &B[\beta]\varphi_1, 0.5),\ (B[\beta]\varphi_2, 0.6),\ (B[\beta](\varphi_1 \wedge \varphi_2), 0.3)\} \end{aligned}$$

Then, using Bridge rule (3) and choosing the function f as

$$f(d, r, c) = r \cdot (1 - c + d)/2,$$

which computes an expected utility (taking the value $(1 - c + d)/2$ as the global utility of reaching a goal with desire degree d and cost c, and 0 otherwise), the agent computes the different intention degrees towards the goals by considering the different feasible plans (i.e. α or β). In this example, the intention degrees for the goal with the highest desire degree, i.e. $\varphi_1 \wedge \varphi_2$, are:

$$(I_\alpha(\varphi_1 \wedge \varphi_2), 0.273) \text{ and } (I_\beta(\varphi_1 \wedge \varphi_2), 0.210)$$

From these results, the *assistant agent* choses to recommend Peter the plan α that would allow him to attend the conference and to visit his friend $(\varphi_1 \wedge \varphi_2)$.

3 Process Calculus

Process calculi have been used to cope with formal aspects of multi-agent interactions. We recall some of these calculi below.

The π-calculus is a process calculus developed by Milner et al. [23] as a continuation of the body of work on the process calculus CCS (Calculus of Communicating Systems) [24]. 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. [6] 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 Subsection 3.1. The Lightweight Coordination Calculus (LCC) [29] 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 interactions 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 to send or to receive messages, where sending a 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 connectives 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 [34]. Walton in [36] presents a language based on CCS [24] to specify agent protocols in a flexible manner during the interaction of agents. Then, in [33, 35] 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 [21] 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) provides 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 with the aim of dealing with the internal structure of intentional agents. We focus on the work about Ambient Calculus [6] to capture the notion of bounded ambient and we take into account some elements of LCC syntax [29] to represent the

state components (e.g. terms, variables).

3.1 Mobile Ambient Calculus

This section is intended to be a refresher of those main notions and concepts of ambient calculus that will be relevant for our purposes, for more details the reader is referred e.g. to [6, 23].

Ambient calculus was developed as a way to express mobile computation [6]. It can also be viewed as an extension of the basic operators of the π-calculus [23]. 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. Ambient calculus (informally) addresses this problem by defining an ambient 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 is by shared locations within a common boundary. Ambients can also be nested, leading to a determined 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.

The syntax of Ambient calculus is shown in Table 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 1. Syntax of Ambient calculus

The syntactic categories are *processes (P, Q, and R)* that includes both ambient

and agents that execute actions, and *capabilities (M)*. Informally the semantics of ambients is as follows. A process is analogous to an individual agent. A process may be one that does nothing (0), may be placed inside an ambient ($M\,[P]$), may be replicated ($!P$), and may be composed in parallel with another process ($P|Q$), which means that the processes execute together. The restriction operator ($(\nu n)\,.P$) creates a new (unique) name n within a scope P.

In Ambient calculus, $n[P]$ denotes an ambient named n containing the process P. In $n[P]$ is understood that P is actively running, and P can be the parallel composition of of several processes.

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 sub-ambients. We say that these processes are running in the ambient, in contrast to the ones running in sub-ambients. The general shape of an ambient is, therefore:

$$n\,[P_1\ |\cdots|\ P_k\ |m_1\,[...]\,|\cdots|m_r\,[...]]$$

To illustrate this structure we may display ambient brackets as boxes as it is shown in Figure 2.

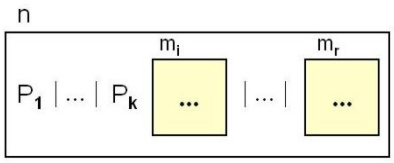

FIG. 2. General structure of an ambient

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.

For further information on the formal definition of Ambient Calculus the reader is referred to [6]. Synthesizing, we can say that the emphasis of this 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.

4 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 [16]. 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 in the MCS. 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 and check whether particular formulae are satisfied and if that is the case, to add a formula in another context ambient. Thus, this process will be getting into and getting out from 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 this calculus a process gets into or out of an ambient C with the ambient enclosing it. In MCC 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:

$$m\,[in\ n.P \parallel n\,[Q]] \ \rightarrow\ m\,[n\,[P \parallel Q]]^2$$

$$m\,[n\,[out\ n.P \parallel R]] \ \rightarrow\ m\,[P \parallel n\,[R]]$$

Furthermore, for defining the MCC calculus we use some elements of the LCC calculus [29] such as the concept of structured terms as constitutive elements of an ambient state. In LCC, terms are used to specify constraints that restrict the interchange of messages and to represent some postconditions after the message sending.

[2] 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.

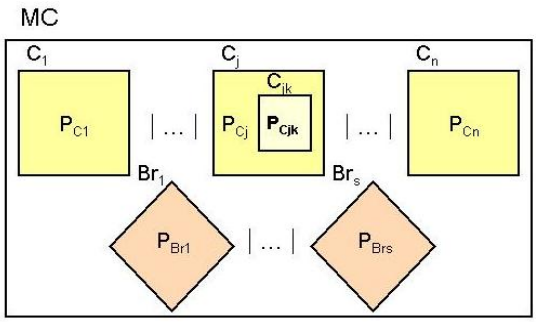

FIG. 3. The general ambient structure for a MCS

In our calculus, the ambient state formulae 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).

The conceptual background of MCC is the global multi-context ambient structure MCS, 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 composing a nested structure. Besides, a bridge rule ambient has an internal state to record substitutions and a special process representing the inter-context deduction, attached to it. The ambient structure for such a MCS is illustrated in Figure 3.

The definition of the MCC syntax is shown in Table 2. In the following items we describe the principal syntax categories in the definition.

Multicontext 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 2). *Clause* leads us to a set of two types 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 (e.g. concurrency problems) between them.

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 the valid formulae in the context (7). In many cases it may be useful to use a nested structure of context ambients. As 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

MCS	::=	$Id_{MC}\ [Clause]$	(1)
$Clause$	::=	$(Clause_c \parallel Clause) \mid (Clause_{br} \parallel Clause)\ \epsilon$	(2)
$Clause_c$	::=	$C\ [P_c \parallel Clause_c] \parallel Clause_c \mid \epsilon$	(3)
$Clause_{br}$	::=	$Br\ [P_{br}]$	(4)
C	::=	$c(Id_c,\ S_c)$	(5)
Br	::=	$br(Id_{br},\ S_{br})$	(6)
S_c	::=	$\{Term\}$	(7)
S_{br}	::=	L	(8)
L	::=	$\langle U \rangle$	(9)
U	::=	$\langle V \rightarrow Term \rangle$	(10)
V	::=	$variable$	(11)
P_c	::=	$Clause_c \mid \vdash_c \mid P_c \cdot P_c \mid P_c\ or\ P_c \mid$ $if\ Term\ then\ P_c\ else\ P_c \mid Action$	(12)
$Action$	::=	$in\ C \mid out\ C \mid get^*(Term, L) \mid$ $getS(L_1, ..., L_n, L) \mid$ $add^*(L, Term) \mid remove(C, Term) \mid \epsilon$	(13)
P_{br}	::=	$Clause_{br} \mid (spy(Br, C_1, \varphi_1, L_1) \parallel$ $spy(Br, C_2, \varphi_2, L_2) \parallel \ldots$ $\parallel spy(Br, C_n, \varphi_n, L_n)) \cdot put^*(Br, C_k, \varphi_k, L_1, ..., L_n)$	(14)
$spy(Br, C, Term, L)$	::=	$out\ Br \cdot in\ C \cdot get^*(Term, L) \cdot out\ C \cdot in\ Br$	(15)
$put^*(Br, C, Term, L_1, ..., L_n)$	::=	$out\ Br \cdot in\ C \cdot getS(L_1, ..., L_n, L) \cdot$ $add^*(L, Term) \cdot\ revise(C) \cdot\ out\ C \cdot in\ Br$	(16)

TABLE 2. Syntax of Multi-context Calculus (MCC)

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 first process is to be executed, otherwise, the second 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 the context ambient state S_c may be redefined as $S_c =:: \{(Term, t)\}$, where the parameter t will be only 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.

5 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 [6], or using rewriting rules for the clause expansion [29]. We have chosen the *natural semantics* 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 specify the semantics of Multi-Agent Protocols (MAP) [33, 35]. In natural semantics we define relations between the initial and final states of program fragments. Thus, we found it suitable for our case since the different processes may

change the ambient states. 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. $C' = c(Id_c, S'_c)$.

$$\frac{C \diamond P_{c1} \Rightarrow C' \quad C' \diamond P_{c2} \Rightarrow C''}{C \diamond P_{c1} \cdot P_{c2} \Rightarrow C''} \tag{5.1}$$

$$\frac{C \diamond P_{c1} \Rightarrow C'}{C \diamond P_{c1} \ or \ P_{c2} \Rightarrow C'} \tag{5.2}$$

$$\frac{C \diamond P_{c1} \Rightarrow C \quad C \diamond P_{c2} \Rightarrow C''}{C \diamond P_{c1} \ or \ P_{c2} \Rightarrow C''} \tag{5.3}$$

$$\frac{C \vdash Term \quad C \diamond P_{c1} \Rightarrow C'}{C \diamond \ if \ Term \ then \ P_{c1} \ else \ P_{c2} \Rightarrow C'} \tag{5.4}$$

Notice that $C \vdash Term$ represents that $Term$ is a valid formula in the ambient state S_c, i.e. $Term \in S_c$.

$$\frac{C \nvdash Term \quad C \diamond P_{c2} \Rightarrow C''}{C \diamond \ if \ Term \ then \ P_{c1} \ else \ P_{c2} \Rightarrow C''} \tag{5.5}$$

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 $add^*(L, Term)$, to define their semantics it is enough to have the 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.

$$\frac{\forall\, Term_i \; \left\{ \left\{ \begin{array}{l} C \vdash Term_i \\ unify(Term, Term_i) \;=\; U_i \end{array} \right\} \leftrightarrow member(U_i, L') \right\}}{\emptyset \;\diamond\; get^*(Term, L) \Rightarrow \;\; L = L'} \tag{5.6}$$

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.

$$\frac{\begin{array}{l} \forall\, (U_1 \in L_1, \ldots, U_n \in L_n) \\ \{(unify^*(U_1, \ldots, U_n) \;=\; L^*) \leftrightarrow member(L^*, L')\} \end{array}}{\emptyset \;\diamond\; getS(L_1, \ldots, L_n, L) \Rightarrow \;\; L = L'} \tag{5.7}$$

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.

$$\frac{\begin{array}{l} C = c(Id_c, S_c) \\ \forall L_i \,\{\; member(L_i, L) \leftrightarrow (Term\,[L_i] \in TermSet)\} \end{array}}{C \diamond \;\; add^*(L, Term) \Rightarrow \;\; c(Id_c, S_c \cup TermSet)} \tag{5.8}$$

Intuitively, $add^*(L, Term)$ adds to the ambient state S_c all the instances of the $Term$ formula by applying the substitutions in L.

6 Mapping a g-BDI Agent to the MCC

Given a g-BDI agent defined by its multi-context specification (see Section 2)

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

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 $\mathcal{A}_g$ in MCC:

$$\mathcal{F} :\;\; A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br}) \;\; \mapsto \;\; \mathcal{A}_g\,[Clause]$$

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. Ambient Calculus enables us to represent nested ambient structures as $C_i\,[P_{C_i} \parallel C_{i0}\,[P_{C_{i0}} \parallel [C_{i1} \parallel ...]]\,]$, which is very useful in order to represent complex

contexts. For example, different ambients may help to individualize different layers used in the context language definition and also in the deduction process.

$$\mathcal{F}:\ C_i = \langle \mathcal{L}_i, A_i, \Delta_i, T_i \rangle \ \mapsto\ c(C_i, S_{C_i})\,[P_{C_i} \parallel C_{i0}\,[P_{C_{i0}} \parallel [C_{i1} \parallel ...]]\ ]$$

- **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, a more basic language it has. The ambient state will be composed by formulae of the top level language (i.e., ambient). This structure allows us to differentiate the language layers in different ambient states, but 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 6.1.

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.:
  $L_i = \langle \langle L_{i1} \rangle, \ldots, \langle L_{in} \rangle \rangle$ where each sublist $\langle L_{ij} \rangle$ 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 2) to represent the bridge rule inference. For the bridge rule Br_i this

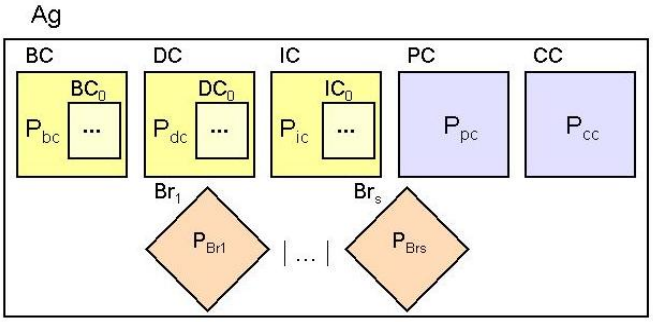

FIG. 4. The ambient structure for a g-BDI agent

process will add determined instances of the formula φ_k in an ambient C_k when the preconditions are satisfied.

$$P_{Br_i} \quad ::= \quad (spy(Br, C_1, \varphi_1, L_1) \parallel \ldots \parallel spy(Br, C_n, \varphi_2, L_n)) \cdot \\ \cdot \; put^*(Br, C_k, \varphi_k, L_1, ..., L_n)$$

The ambient structure in MCC for representing a g-BDI agent A_g is illustrated in Figure 4. Therefore, 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 context are based in first order logic, the mapping is straightforward and both contexts can be easily passed to a corresponding 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 may be developed.

6.1 Mapping the Desire context (DC) into a Desire ambient

We next define a mapping $\mathcal{F}$ from a desire context DC to a suitable ambient structure in MCC

$$\mathcal{F} : DC = \langle \mathcal{L}_{DC}, A_{DC}, \Delta_{DC}, T_{DC} \rangle \;\mapsto\; c(DC, S_{DC})\,[P_{DC} \parallel ...]$$

To do this, we start with a synthesized description of the components of the **Desire Context (DC)**: the language $\mathcal{L}_{DC}$, the axioms A_{DC}, the inference rules Δ_{DC} and a theory T_{DC}. A more complete description of these logical elements can be found in [12].

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

means of the trick of 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 RPL logic (Łukasiewicz logic, extended with rational truth-constants, see Annex I), 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 $\Phi, \Psi \in DC$ then $\Phi \to_L \Psi \in \mathcal{L}_{DC}$ and $\neg_L \Phi \in \mathcal{L}_{DC}$ (where $\neg_L$ and $\to_L$ 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 $\to_L$, see Annex I)

Since in Łukasiewicz 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 $\overline{r} \to_L D^+\varphi$ (respectively $\overline{r} \to_L D^-\varphi$) express that the positive (respectively negative) desire of φ is at least r.

Axioms and inference rules (A_{DC} and Δ_{DC}): to axiomatize the logical system DC we need to combine axioms of classical propositional calculus (CPC) for formulas of $\mathcal{L}$ with Łukasiewicz logic axioms for modal formulae (described in Annex I), plus additional axioms characterizing the behavior of the modal operators D^+ and D^-. The intuitions behind these axioms and rules are that a conjunctive combination of desires of one kind (either positive or negative) may result in a strictly higher desire value (captured by the introduction rules), while the desire value of a disjunctive combination of either positive or negative desires is taken as the minimum of the desire degrees (represented by Ax_1 and Ax_2 respectively):

- Axioms of CPC for formulae of $\mathcal{L}$
- Axioms of Rational Pavelka logic for modal formulae (see Annex I).
- Axioms[3] for D^+ and D^- over Łukasiewciz logic:
  $Ax_1: \ D^+(\varphi \vee \psi) \equiv_L D^+\varphi \wedge_L D^+\psi$
  $Ax_2: \ D^-(\varphi \vee \psi) \equiv_L D^-\varphi \wedge_L D^-\psi$
- Rules are: modus ponens for $\to$ and $\to_L$ and introduction of D^+ and D^- for implications:
  Δ_1: from $\varphi \to \psi$ derive $D^+\psi \to_L D^+\varphi$ and
  Δ_2: from $\varphi \to \psi$ derive $D^-\psi \to_L D^-\varphi$.

Theory (T_{DC}): it consists of a set of formulas from $\mathcal{L}_{DC}$

[3] These are the basic ones, one could consider additional axioms introducing some constraints between both modalities.

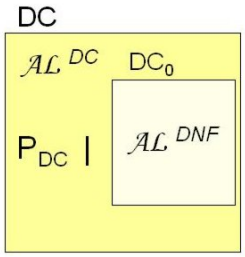

FIG. 5. The DC ambient structure

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}.

1. **State Language** $\mathcal{AL}^{DC}$
   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: $DC\ [P_{DC} \parallel DC_0]$. This structure and the languages involved are illustrated in Figure 5.
   This nested ambient structure enables 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** S_{DC}
   The DC ambient state S_{DC} is composed by the translated formulae of the context theory: $S_{DC} = \{\mathcal{F}(\Phi) \mid \Phi \in T_{DC}\}$

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 since there exist theorem provers for this many-valued logic [2]. 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 be applied. Thus, the P_{DC} process is defined as the following schema of sequential processes:

$$P_{DC} \ ::= \ P_V(VSet) \cdot P_{DNF}(VSet) \cdot P_{AX} \cdot P_\Delta \cdot P_{\text{Ł}}$$

In the following items we describe the four first processes:

(P1) The $P_V(VSet)$ process extracts from S_{DC} the finite set of variables appearing in the formulas of T_{DC} and puts them in $VSet$.

(P2) The $P_{DNF}(VSet)$ process enters in the DC_0 ambient and through the process $add^*_{DNF}(VSet)$ creates and adds to S_{DC_0} the finite set of DNF formulae built upon the variables in $Vset$, i.e.:

$$P_{DNF}(VSet) ::= in \ DC_0 \cdot add^*_{DNF}(VSet) \cdot out \ DC_0$$

(P3) 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 represent 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 represents the corresponding axiom for D^-.

(P4) 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

$$P_\Delta ::= P_{\Delta_1} \cdot P_{\Delta_2} \ , \qquad \text{with}$$

$$P_{\Delta_1} ::= in\ DC_0 \cdot\ get^*(\mathcal{F}(x \to y), L) \cdot\ out\ DC_0 \cdot\ add^*(\mathcal{F}(D^+(y) \to D^+(x)), L)$$

and similarly for P_{Δ_2}, where

– $get^*(F(x \to y), L)$ gets the pairs (x, y) resulting from the unification of $F(x \to y)$ in DC_0 ambient;

– $add^*(\mathcal{F}(D^+(y) \to D^+(x)), L)$ adds all the instances of the formula $\mathcal{F}(D^+(y) \to D^+(x))$ with pairs $(x, y) \in L$.

(P5) The final process P_L applies Łukasiewicz logic deduction $\vdash_L$ to the state S'_{DC} resulting from the previous processes, i.e. $P_L ::= \vdash_L$

The complexity of processes P1 and P4 is $O(n)$, considering n as the size of the theory T_{DC}. Process P2 can be $O(2^n)$ in the worst case and process P3 is quadratic on n, $O(n^2)$. The difficult process is the last one, P5, that is a co-NP complete problem [1] and thus can lead to exponential computations in the worst case. The belief contexts have a similar complexity and the intention context acts as a kind of data base recording the results of bridge rules and thus its complexity is basically lineal on the size of the theory, $O(n)$. Planning is, even in simple representations like STRIPS, NP-Complete. Thus, the planner context will be in general exponential on the size of its theory (based on goals and beliefs).

7 Discussions

In this work we cope with the operational semantics aspects of the g-BDI agent model. 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 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 agent model has been used to specify and implement concrete recommender agents in the tourism domain [10]. The agent performance may be improved by using these graded attitudes as it was shown in the experimentation of this case study (see [11]). 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 [13].

For giving computational meaning to our agent model, we have first defined a MCC Calculus for Multi-context systems (MCS) execution. The MCC proposed is based on Ambient calculus [6] and includes some elements of LCC [36]. The operational semantics for this language was given using Natural Semantics. MCSs have been used in diverse applications as for example in the integration of heterogeneous knowledge and data bases, in the formalization of reasoning about propositional attitudes [16] and to engineer agents in multiagent systems [26]. Thus, 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. Through MCC we have given to this agent model computational semantics 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.

Acknowledgments The authors are thankful to the anonymous reviewers for their helpful comments for improving the paper. Ana Casali acknowledge partial support by the PID-UNR ING308 project. Lluís Godo and Carles Sierra acknowledge partial support by the Spanish project *Agreement Technologies* (CONSOLIDER CSD2007-0022, INGENIO 2010).

References

[1] Aguzzoli S., Gerla B. and Haniková Z. Complexity Issues in Basic Logic, *Soft Computing*, 9, pp. 919-934, 2005.

[2] Beavers G. Automated theorem proving for Łukasiewicz logics. *Studia Logica*, Volume 52, Number 2 pp. 183-195, 1993.

[3] 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.

[4] Brewka G., Roelofsen F., and Serafini L. Contextual Default Reasoning. In Proc. of IJCAI 2007, pp. 268273.

[5] Dellunde P. and Godo L. Introducing Grades in Deontic Logics. In Proc. of DEON'08. Lecture Notes in Computer Science vol. 5076, Ron van der Meyden and Leon van der Torre (eds.), Springer-Verlag, pp. 248-262, 2008.

[6] Cardelli L. and Gordon A.D. Mobile Ambients. In Maurice Nivat, editor, *Foundations of Software Science and Computational Structures*, number 1378 in Lecture Notes in Computer Science, pp. 140-155. Springer-Verlag, 1998.

[7] Casali A. On intentional and social agents with graded attitudes, *Monografies de l'Institut d'Investigació en Intel.ligència Artificial*, CSIC, 2009.

[8] 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*, pp. 126-143, Springer-Verlag, Berling Heidelberg, 2005.

[9] 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.

[10] Casali A., Godo L. and Sierra C. Modeling Travel Assistant Agents: a graded BDI Approach. IFIP, Volume 217, *Artificial Intelligence in Theory and Practice*, Ed. Max Bramer (ISBN 0-387-34654-6) (Boston: Springer), 415-424, 2006.

[11] 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.

[12] 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*, G. Brewka and J. Lang (Eds.), The AAAI Press, pp.27-37, 2008.

[13] Deco C., Bender C., Casali A., Motz R. Design of a Recommender Educational System. Proceedings of *3ra. Conferencia Latinoamericana de Objetos de Aprendizaje (LACLO 2008)*, Aguascalientes, Mexico, 2008.

[14] Dunin-Keplicz B. and Verbrugge R. *Teamwork in Multi-Agent Systems: a formal approach*. Wiley, 2010.

[15] Gabbay D.M. *Labelled Deduction Systems*, Volume 1, Oxford Logic Guides, Volume 33, Clarendon Press / Oxford Science Publications, 1996.

[16] Ghidini C. and Giunchiglia F. Local Model Semantics, or Contextual Reasoning = Locality + Compatibility *Artificial Intelligence*,127(2):221-259, 2001.

[17] Godo L., Esteva F. and Hájek P. Reasoning about probabilities using fuzzy logic. *Neural Network World*, 10:811–824, 2000.

[18] 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.

[19] Hájek P., Metamathematics of Fuzzy Logic, *Trends in Logic*, 4, Kluwer Academic Publishers (1998).

[20] Halpern, J.Y. *Reasoning About Uncertainty*, MIT Press, Cambridge, MA, 2003.

[21] Joseph S., Perreau de Pinninck Bas A., Robertson D., Sierra C. and Walton C. Interaction Model Language Definition. *IJCAI 2007 Workshop AOMS Agent Organizations Models and Simulations*. Dignum V., Dignum F., Matson E. and Edmonds B. eds., pp. 49-61, 2007.

[22] Lorini E., Herzig A., Broersen J. and Troquard N. Grounding Power on Actions and Mental Attitudes, Formal Aspects of Multiagent Systems, *Logic Journal of the IGPL*, Verbrugge R. and Keplicz B. guest eds. This volume.

[23] Milner R., Parrow J. and Walker D. A calculus of mobile processes, Parts 1-2. *Information and Computation*, 100(1), 1-77. 1992.

[24] Milner R. *Communication and Comcurrency*. Prentice-Hall International, 1989.

[25] Ognjanovic Z. and Raskovic M. Some Probability Logics with New Types of Probability Operators. *Journal of Logic and Computation* 9(2): 181-195, 1999.

[26] 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.

[27] Pinyol I. and Sabater-Mir J. Integrating Image and Reputation Information in BDI Agents In Proc. of *EUMAS 08*, Bath,UK, 2008.

[28] 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.

[29] Robertson D. Multi-Agent Coordination as Distributed Logic Programming. In Bart Demoen, Vladimir Lifschitz (Eds.): *Logic Programming, 20th International Conference, ICLP 2004*, Saint-Malo, France, September 6-10, 2004, Proceedings. Lecture Notes in Computer Science, Springer, pp. 3132, pp. 416-430.

[30] Sabater J., Sierra C., Parsons S. and Jennings N. R. Engineering executable agents using multi-context systems. *Journal of Logic and Computation* 12(3), pp. 413-442, 2002.

[31] van der Hoek, W. On the Semantics of Graded Modalities. *Journal of Applied Non-Classical Logics* 2(1), pp. 81-123, 1992.

[32] van Ditmarsch, H. Prolegomena to Dynamic Logic for Belief Revision, *Synthese* 147, pp. 229-275, 2005.

[33] Walton C. Multi-Agent Dialogue Protocols. In Proceedings of the *Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, January 2004.
[34] Walton C. Model Checking Multi-Agent Web Services. In Proceedings *AAAI Spring Symposium on Semantic Web Services*, Stanford, California, 2004.
[35] Walton C. Verifiable agent dialogues. *Journal of Applied Logic* 5, pp. 197-213, 2007.
[36] Walton C. and Robertson D. Flexible multi-agent protocols. Presented at UKMAS 2002, Liverpool, UK, December 2002. Published as *Informatics Technical Report EDI-INF-RR-0164*, University of Edinburgh, December 2002.

Annex I: Rational Pavelka Logic

Rational Pavelka logic RPL is an extension of Łukasiewicz's infinitely-valued logic admitting to explicitly reason about partial degrees of truth. Introduced by Pavelka in the late seventies, it is described in a simple formalization in [19]. Since the approach described in this paper strongly relies on this logic, here we follow the latter and present its main notions and properties.

Formulae are built from propositional variables $p_1, p_2, \ldots$ and truth constants $\overline{r}$ for each *rational* $r \in [0,1]$ using connectives $\to_L$ and $\neg_L$. Other connectives can defined from these ones. In particular, among others, one can define two conjunctions and two disjunctions exactly as in Łukasiewicz's logic, i.e.

$$\begin{array}{rcl}
\varphi \otimes \psi & \text{stands for} & \neg_L(\varphi \to_L \neg_L \psi) \\
\varphi \oplus \psi & \text{stands for} & \neg_L \varphi \to_L \psi \\
\varphi \ominus \psi & \text{stands for} & \neg_L(\varphi \to_L \psi) \\
\varphi \vee_L \psi & \text{stands for} & (\varphi \to_L \psi) \to_L \psi \\
\varphi \wedge_L \psi & \text{stands for} & \neg_L(\neg_L \varphi \vee_L \neg_L \psi) \\
\varphi \equiv_L \psi & \text{stands for} & (\varphi \to_L \psi) \wedge_L (\psi \to_L \varphi)
\end{array}$$

Łukasiewicz's truth functions for the connectives $\to_L$ and $\neg_L$ are (we use the same symbol as for the connectives):

$$\begin{array}{rcl}
x \to_L y & = & \min(1, 1 - x + y) \\
\neg_L x & = & 1 - x
\end{array}$$

Taking into account these definitions, it is easy to check that the truth functions for the above definable connectives are the following ones:

$$\begin{array}{rcl}
x \otimes y & = & \max(0, x + y - 1) \\
x \oplus y & = & \min(1, x + y) \\
x \ominus y & = & \max(0, x - y) \\
x \vee_L y & = & \max(x, y) \\
x \wedge_L y & = & \min(x, y) \\
x \equiv_L y & = & 1 - |x - y|
\end{array}$$

An *evaluation* e is a mapping of propositional variables into $[0,1]$. Such a mapping uniquely extends to an evaluation of all formulae respecting the above truth functions and further assuming that $e(\overline{r}) = r$ for each rational $r \in [0,1]$. An evaluation is a model of a set of formulae $\mathcal{T}$ whenever $e(\varphi) = 1$ for all $\varphi \in \mathcal{T}$. We write $\mathcal{T} \models_{RPL} \varphi$ to denote that $e(\varphi) = 1$ for every evaluation e model of $\mathcal{T}$.

Logical axioms of RPL are:
(i) axioms of Łukasiewicz's logic

$\varphi \to_L (\psi \to_L \varphi)$
$(\varphi \to_L \psi) \to_L ((\psi \to_L \chi) \to_L (\varphi \to_L \chi))$
$(\neg_L\varphi \to_L \neg_L\psi) \to_L (\psi \to_L \varphi)$
$((\varphi \to_L \psi) \to_L \psi) \to_L ((\psi \to_L \varphi) \to_L \varphi)$

(ii) bookkeeping axioms: (for arbitrary rationals $r, s \in [0, 1]$):

$\neg_L\overline{r} \equiv_L \overline{1 - r}$
$\overline{r} \to_L \overline{s} \equiv_L \overline{\min(1, 1 - r + s)}$

The only *deduction rule* is modus ponens for $\to_L$. The notion of proof in RPL, denoted $\vdash_{RPL}$, is defined as usual from the above axioms and rule.

RPL enjoys two kinds of completeness. The so-called Pavelka completeness reads as follows. Let $\mathcal{T}$ be an arbitrary set of formulae (theory) and φ a formula. The *provability degree* of φ in $\mathcal{T}$ is defined as $|\varphi|_T = \sup\{r \mid \mathcal{T} \vdash_{RPL} \overline{r} \to_L \varphi\}$. The *truth degree* of φ in $\mathcal{T}$ is defined as $\|\varphi\|_T = \inf\{e(\varphi) \mid e \text{ evaluation}, e \text{ model of } \mathcal{T}\}$. Notice that both $\|\varphi\|_T$ and $|\varphi|_T$ may be irrational.Then, for each $\mathcal{T}$ and φ, it holds that

$$|\varphi|_T = \|\varphi\|_T,$$

i.e. the provability degree equals to the truth degree.

Besides, RPL enjoys a classical completeness property but only for deductions from finite theories, which will be used in the proof in Annex II, and reads as follows: for each *finite* $\mathcal{T}$ and φ, $T \vdash_{RPL} \varphi$ iff $T \models_{RPL} \varphi$.