# Using Jason to Develop Normative Agents

Baldoino F. dos S. Neto[1], Viviane T. da Silva[2], Carlos J. P. de Lucena[1]
{bneto, lucena}@inf.puc-rio.br, and
viviane.silve@ic.uff.br

[1] PUC-Rio, Informatics Department, LES, Rio de Janeiro - Brazil
[2] Universidade Federal Fluminense, Computer Science Department, Niteri - Brazil

**Abstract.** Norms have become one of the most promising mechanisms of social control to ensure a desirable social order in open multi-agent systems where autonomous, heterogeneous and independently designed entities can work towards similar or different ends. Norms regulate the behaviour of agents by defining permissions, obligations and prohibitions, and by stating stimulus to their fulfilment while defining rewards and discouraging their violation while pointing out punishments. Since goal-oriented agents' priority is the satisfaction of their own desires, they must evaluate the positive and negative effects of the fulfilment or violation of the norms before choosing to comply or not with them. In this context, we present the new functions of the Jason platform defined to support normative reasoning, i.e, to build agents able to deal with desires and norms. Agents are then able to check if they should adopt or not the norm, evaluate the effects of the fulfilment or violation of the norm on their desires, detect and solve conflicts among norms, and select desires and plans according to their choices of fulfilling or not a norm. We demonstrate the applicability of such new functions through a non-combatant evacuation scenario.

## 1 Introduction

Open multi-agent systems are societies in which autonomous, heterogeneous and independently designed entities can work towards similar or different ends [7]. In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, those systems establish a set of norms that is used as a mechanism of social control to ensure a desirable social order in which agents work together [7].

Such norms regulate the behaviour of the agents by defining obligations (indicating that agents are obligated to accomplish something in the world), permission (indicating that agents are permitted to act in a particular way) and prohibitions (indicating that they are prohibited to act in a particular way)[9]. Moreover, norms may give stimulus to their fulfilment by defining rewards and may discourage their violation while stating punishments [12].

Over the last years, several approaches have been proposed on the specification and implementation aspects of norms, such as [4] and [12]. Others have focuses on the definition of parts of an infrastructure to be used by BDI agents [11] to reasoning on norms, such as [8] [6]. However, there is still a need to define an agent-oriented platform able to guide the implementation of goal-oriented normative agents, i.e., agents that have the main purpose of achieving their desires while trying to fulfil the system norms. From the set of main used agent-oriented platform such as [1] [5], none provides support to build normative agents.

In this context, we present the new functions of the Jason platform [1] defined to support normative reasoning, i.e, to build agents able to deal with desires and norms. The original Jason platform already provides support to the implementation of BDI agents and a set of hot-spots that enable the implementation of normative functions. By using the new functions being proposed, it is possible to build BDI agents able to check if they should adopt or not a norm, evaluate the effects, on their desires, of the fulfilment or violation of the norm, detect and solve conflicts among norms, and select desires and plans according to their choices of fulfilling or not a norm.

We demonstrate the applicability of the new functions we have defined through a non-combatant evacuation scenario where the asks related to adopt, evaluate, and comply norms are shown.

The paper is structured as follows. In Section 2 we outline the background about norms. In Section 3 the Jason platform is explained, in Section 4 the scenario used to present the work is presented and in Section 5 we present the normative Jason platform and use the scenario to exemplify it. Section 6 summarizes relevant related work and, finally, Section 7 concludes and presents some future work.

## 2 Norms

In this work, we follows the norm representation described in [12], as shown bellow:

*norm* (*Addressee*, *Activation*, *Expiration*, *DeonticConcept*, *State*, *Rewards*, *Punishments*) where *Addressee* is the agent or role responsible for fulfilling the norm, *Activation* is the activation condition for the norm to become active, *Expiration* is the expiration condition for the norm to become inactive, *Rewards* are the rewards to be given to the agent for fulfilling a norm, *Punishments* are the punishments to be given to the agent for violating a norm, *DeonticConcept* indicates if the norm states an obligation or a prohibition[3], and *State* describes the set of states being regulated. In this paper we are only dealing with norms that restrict the achievement of a given state. We are not considering norms that directly regulate the execution of actions since in Jason it is not possible to apply unification between internal actions.

## 3 Jason Platform

Jason is an interpreter for an extended version of AgentSpeak proposed by Rao [10] that gives support to the creation of BDI agents. Figure 1[4] (reproduced from [1]) illustrates the MAS platform provided by Jason. Sets (of beliefs, events, plans, and intentions) are represented as rectangles, diamonds are used to represent selection functions (of

[3] In this paper we assume that everything is permitted unless a prohibition is stated

[4] The dark, internal diamonds were defined in the normative version of the platform (NRF, UN, EN, DSC, ADP, Sg, Sp)

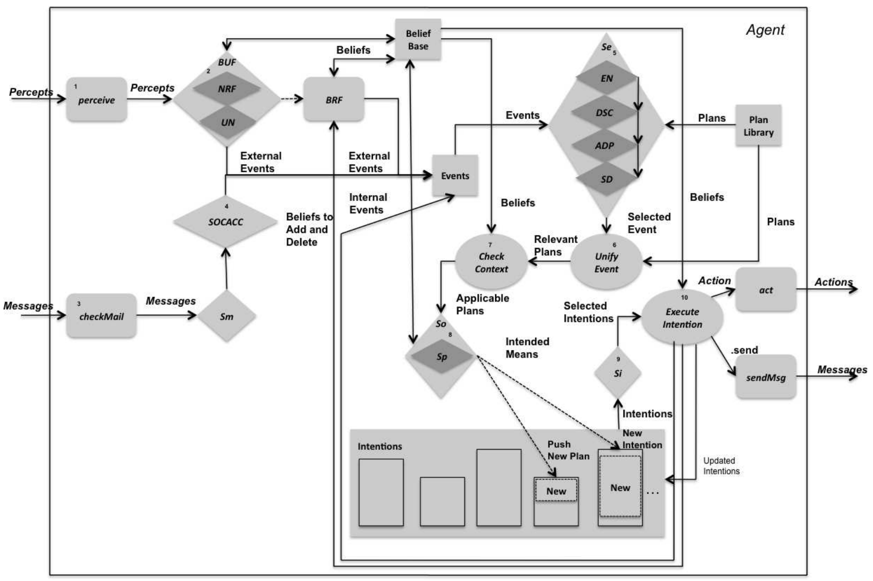

**Fig. 1.** Normative Jason Platform

one element from a set) and circles to represent some of the processes involved in the interpretation of AgentSpeak programs.

Each interpretation cycle updates the list of events according to the perception coming from the environment, to the messages the agent receives and to the information coming from the agent's own execution of a plan. In the *Belief Update Function (BUF)* the perceptions and actual agent's beliefs are used to update the *Belief Base* and to update the set of events to be carried on. From the set of messages received, the *Message Selection Function (SM)* selects one to be handled. The *SocAcc function* can filter the messages based on the characteristics of the sender. The *Belief Review Function (BRF)* revises the *Belief Base* with a literal to be added or deleted, and the intention structure that required the belief change. A single event is selected in the *Event selection function (SE)* that is unified with triggering events in the heads of plans by the *Unify Event cycle* generating a set of all relevant plans. The context of such plans are verified according to the *Belief Base* by the *Check Context cycle* generating a set options. The *Option Select Function (SO)* selects a single applicable option from the set of options, which becomes the intended means for handling the selected event. The option either pushes the plan on the top of an existing intention (if the event was an internal one), or creates a new intention in the set of intentions (if the event was external, i.e., generated from perception of the environment). The *Intention Select Function (SI)* selects one of the agent's intentions that is executed by the *Execute Intention cycle*. When all formula in the body of a plan have been executed, the whole plan is removed from the intention list, and so is the achievement goal that generated it. This ends a cycle of execution, and AgentSpeak starts all over again, checking the state of the environment after agents have acted upon it, generating the relevant events, and so forth.

## 4 Scenario: rescue operation

Our implementation is based in the simplified non-combatant evacuation scenario, presented in [2]. In such scenario agents are responsible to plan the evacuation of members of a Non-Governmental Organisation (NGO) that are in hazardous location. The agents operate in different areas with different resources, such as: *(i)* different Autonomous Unmanned aerial Vehicles (AUVs), deployed with sensors to provide information about the enemies like types of weapons used by them, location of their bases and strategies, and information about the area operated by each agent like information about the weather, *(ii)* helicopters, *(iii)* troops and *(iv)* land-based helicopters. Considering that such resources are limited, we have a *Commander Agent* that is responsible to regulate the behaviour of the agents and the use of the resources according to the following norms:

**Norm1**

**Addressee**: *Rescue Entity*
**Activation**: NGO workers are stranded in a hazardous location
**Expiration**: NGO workers are stranded in a safe location
**DeonticConcept**: Obligation
**State**: To evacuate NGO workers
**Rewards**: The *Commander Agent* gives more troops to *Rescue Entity*.
**Rewards**: The *Commander Agent* gives land-based helicopters to *Rescue Entity*.
**Punishments** : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their troops.

**Norm 2**

**Addressee**: *Rescue Entity*
**Activation**: The weather is bad
**Expiration**: The weather is good
**DeonticConcept**: Prohibition
**State**: To evacuate NGO workers
**Punishments** : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their helicopters. **Punishments** : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their land-based helicopters.

**Norm 3**

**Addressee**: *Rescue Entity*
**Activation**: The weather is bad
**Expiration**: The weather is good
**DeonticConcept**: Prohibition
**State**: To use helicopters.
**Rewards**: The *Commander Agent* gives more troops to *Rescue Entity*.
**Rewards**: The *Commander Agent* gives land-based helicopters to *Rescue Entity*.
**Punishments** : (obligation) *Rescue Entity* is obligated to return to the *Commander Agent* part of their troops.

## 5 The Normative Jason Platform

The implementation of the Jason platform proposed in this paper aims to help agents on reasoning about the system norms. Norm is considered a primary concept that influences the agent while reasoning about its beliefs, desires, plans and intentions. In a nutshell, the extended Jason platform modifies the original Jason platform by including the following functions[5], as illustrated in Figure 1: (Norm Review Function - **NRF**) this function helps the agent on recognizing its responsibilities towards other agents by incorporating the norms that specify such responsibilities. That is, the main goal of this function is to update the set of adopted norms taking into accounting the perceptions and the information in the belief base; (Updating Norm - **UN**) after reviewing the adopted norms and the beliefs, some norms' activation conditions and deactivation ones can be trigged. Therefore, the main task of this step is dedicated to update the set of activated and adopted norms; (Evaluating Norm - **EN**) this step helps the agent on selecting, from the set of activated norms, the norms that it has the intention to fulfil and the ones it has the intention to violate; (Detecting and Solving Conflicts - **DSC**) this function checks and solves the conflicts among the norms; (Annotating Desires and Plans- **ADP**) after deciding the norms to be fulfilled, the desires and the plans are annotated with a priority level. The desires that positively influence the fulfilment of the norm and the plans able to fulfil the norms receive highest priority; (Selecting Desires - **SD**) the main goal of this step is to select the desires that will become intentions taking into according their priorities. By default the desire with highest priority is selected; (Selecting Plans - **SP**) this function chooses a single applicable plan from the *Plan Library* according to their priorities. By default the plan with highest priority is selected.

## 5.1 Norm Review Function (NRF)

This function recognizes from the set of receiving perceptions the ones that describe norms. After recognizing the norms, such function reviews the set of adopted norms applying the following verifications: (i) it checks if the new norm unifies with one of the norms already adopted, i.e., if the incoming norm already exists in the agent *Belief Base*, and (ii) it verifies if the agent is the addressee of the norm, i.e., if the field *Addressee* of the new norm unifies with the agent role or agent name, also stored as a belief in the *Belief Base*. Finally, such function updates the set of adopted norms in the *Belief Base* if the new norm does not already exist and the agent is the addressee of the norm.

With the aim to exemplify the use of this function, let's consider the scenario presented in Section 4 where two groups of agents are leaded by *Agent A* and *Agent B* playing the role *Rescue Entity*. When these entities receive information about the three system norms, the NRF function is executed comparing the addressee information with the role being played by the agents and checking if the norms are not stored yet in the agent's belief base.

## 5.2 Updating Norm (UN)

UN function updates the set of activated and adopted norms checking if the fields *Activation* and *Expiration* of the norm unifies with the beliefs of the agent. If the activation conditions unify with the beliefs, the adopted norm is activated. In the expiration conditions unify with the beliefs, the norm is deactivated and stored as an adopted norm. Note that only the norms that are active must be fulfilled.

[5] The normative Jason platform together with its new functions are available at http://wiki.les.inf.puc-rio.br/index.php/NormativeAgent

Following the example above, if the weather of the area operated by one of the two rescue entities is bad, both norms 2 and 3 are activated, since the activation condition of both norms is “The weather is bad”. If the norms are activated, the rescue entity must not rescue NGO workers and must not use helicopters. Both norms are deactivated when the expiration condition unifies with the information about a good weather stored in the agent’s belief base.

## 5.3 Evaluating Norm (EN)

This function evaluates the benefits of fulfilling or violating the norms, i.e., it checks how close the agent gets of achieving its goals if it decides to fulfil or if it decides to violate the norms. In order to do this, the following steps are performed by considering the norms stored in the *belief base*:

1. In case of obligations, i. e., if the field *DeonticConcept* of the norm is equal to obligation, it checks if the state described in the norm is equal to one of the states the agent has the desire to achieve (including the desires stored in the *event library* and the desires that compose the options of the *plan library*). In affirmative cases, it considers that the obligation contributes positively to help the agent on achieving its desires. In any other case, the obligation will not contribute or will contribute negatively.
2. In case of prohibitions, i. e., if the field *DeonticConcept* of the norm is equal to prohibition, it also checks if the state described in the norm is equal to at least one of the states the agent has the desire to achieve. It means that the agent has the desire to achieve a state that it is being prohibited to. In affirmative cases, it considers that the prohibition contributes negatively since it disturbs the achievement of the agent’s desires. In any other case, the prohibition will contribute neutrally.
3. After analysing the state , this step considers the influence that the rewards have to the achievement of the agent’s desires. We consider that a reward can never influence the agent negatively but always positively or neutrally. In case the reward really helps the agent on achieving its desires, its influence is positive but there may be cases that the reward is useless.
4. Finally, the punishments are evaluated in order to check if they will influence the achievement of the agent’s desires negatively or positively.
   a In case the punishment states a change in the agent beliefs, it can influence the agent neutrally or negatively but probably never positively since it is a punishment.
   b In case the punishment states a prohibition and the state being prohibited is one of the agent desires, the punishment will influence negatively since it will disturb the agent of achieving one of its goals. If it is not, the punishment will not influence.
   c In case the punishment states an obligation and the state being obliged is one of the agent desires, the punishment will influence positively (or neutrally) since such state will already be achieved by the agent. Otherwise, the punishment will influence negatively since the agent has not the desire to achieve such state.

Such considerations are summarized in Table 1. Note that instances of this table must be created for each norm in order to individually check its contribution to the achievement of the agent's desires.

**Table 1.** Evaluating the norms. D means that the agent has the desire/intention, C that the rewards contribute to achieve the desire/intention, O that the beliefs obscure the achievement of the desire/intention, and O that there is not a case of such contribution.

| 1 | ***Norm*** | ***Contribution*** | | |
|---|---|---|---|---|
| 2 | | **positive** | **neutral** | **negative** |
| 3 | **Obligation** | D | not D | not D |
| 4 | **Prohibition** | Ø | not D | D |
| 5 | **Reward** | C | not C | Ø |
| 6 | ***Punishments*** | | | |
| 7 | **belief** | Ø | not O | O |
| 8 | **Prohibition** | Ø | not D | D |
| 9 | **Obligation** | D | not D | not D |

In the end, the *EN* function groups the activated norms in two sub-sets: norms to be fulfilled and norms to be violated. Where by default the norm is added to the sub-set *Fulfil* of the activated norms if the contribution for fulfilling the norm is greater than or equal to the contribution for violating the norm. Otherwise, it is selected to be added to the sub-set *Violate*;

In order to exemplify the applicability of this function, let's consider the rescue operation scenario. The evaluation of the benefits of fulfilling and violating the 3 norms are shown in the Tables 2, 4 and 3 that indicates the contribution of each norm element to the achievement of the agent goals. In order to simplify the example, we consider that any norm element generates the same contribution that is 1.

So, from the contribution of the norms shown in the Tables 2, 4 and 3, Norm 1 is included in the set of norms to be fulfilled since the contribution for fulfilling it is equal to "+3" and greater than the contribution for violating it that is equal to "-1". Norm 2 is also included in the fulfil set since the contribution for fulfilling it is equal to "-1" and greater than the contribution for violating it that is equal to "-2". And, finally, Norm 3 is included in the fulfil set since the contribution for fulfilling it is equal to "+1" and greater than the contribution for violating it that is equal to "-1". It indicates that the agent has the intention to fulfil the three norms.

## 5.4 Detecting and Solving Conflicts (DSC)

The goal of the DSC function is to check and solve conflicts between norms. Such conflicts can happen if two different norms (one being an obligation and the other one a prohibition) specify the same state, both norms have been selected to be fulfilled or to be violated and both norms are active at the same time.

If the agent intends to fulfil the obligation but does not intend to fulfil the prohibition, these norms are not in conflict. The same can be said if the agent intends to fulfil

**Table 2.** Evaluating norm 1

| 1 | ***Norm*** | ***Contribution*** | | |
|---|---|---|---|---|
| 2 | | **positive** | **neutral** | **negative** |
| 3 | **Obligation** | 1 | 0 | 0 |
| 5a | **Reward** | 1 | 0 | Ø |
| 5b | **Reward** | 1 | 0 | Ø |
| 6 | ***Punishments*** | | | |
| 9 | **Obligation** | 0 | 0 | 1 |

**Table 3.** Evaluating norm 3

| 1 | ***Norm*** | ***Contribution*** | | |
|---|---|---|---|---|
| 2 | | **positive** | **neutral** | **negative** |
| 4 | **Prohibition** | 0 | 0 | 1 |
| 5a | **Reward** | 1 | 0 | Ø |
| 5b | **Reward** | 1 | 0 | Ø |
| 6 | ***Punishments*** | | | |
| 9 | **Obligation** | 0 | 0 | 1 |

**Table 4.** Evaluating norm 2

| 1 | ***Norm*** | ***Contribution*** | | |
|---|---|---|---|---|
| 2 | | **positive** | **neutral** | **negative** |
| 4 | **Prohibition** | 0 | 0 | 1 |
| 6 | ***Punishments*** | | | |
| 9a | **Obligation** | 0 | 0 | 1 |
| 9b | **Obligation** | 0 | 0 | 1 |

the prohibitions and to violate the obligation. On the other hand, if the agent intends to fulfil both norms or to violate both norms, they are in conflict and it must be solved.

By default, in case of conflicts between two norms that the agent intends to fulfil or violate, this function proposes to select the one with highest contribution to the achievement of the agent's desires. That is, if the contribution coming from the fulfilment of the first norm plus the contribution coming from the violation of the second norm is greater than or equal to the contribution coming from the fulfilment of the second norm plus the contribution coming from the violation of the first norm, the first norm is selected to be fulfilled and the second to be violated.

Considering the norms contribution evaluated in the *EN* function, a conflict between Norm 1 and 2 is detected and should be solved. The conflict is solved by selecting Norm 1 to be fulfilled and Norm 2 to be violated since the contribution coming from the fulfilment of the first norm (+3) plus the contribution coming from the violation of the second norm (-3) is greater than the contribution coming from the fulfilment of the second norm (-1) plus the contribution coming from the violation of the first norm (-4).

## 5.5 Annotating Desires and Plans (ADP)

The ADP function annotates the desires and plans according to the deontic concept associated with it (obligated or forbidden). In order to do this, each desire or plan is annotated with a level of priority following two steps: **(Events Verification)** The main goal of this step is to annotate the desires in the *Event Library* taking into according the norms the agent wants to fulfil. In others words, if the agent has a desire to achieve a state and there is a norm that oblige the agent of achieving such state, the desire priority is increased according to the importance of the norm. If the agent has a desire to achieve a state and there is a norm that prohibit the agent of achieving such state, the desire priority is decreased according to the importance of the norm. If there is not any norm related to the desires, its priority is not modified. For example, considering that a “Rescue Entity” has the intention to fulfil Norm 1, the desire “Evacuating the stranded workers to a safe location” is annotated with priority equal to 1.**(Plans Verification)** In this case, if the state described by an obligation norm is equal to one of the states achieved by the plan, the norm increases the priority of such plan and if the state described by a prohibition norm is equal to one of the states of the plan, the norm decreases the priority of such plan. For example, considering that a “Rescue Entity” has intention to fulfil the Norm 3, the priority of plans that uses helicopters are decreased of -1.

### 5.6 Selecting Desires (SD)

The SD function is responsible for selecting the desires with highest priority. By applying this function to our example, the goal “Evacuating the stranded workers to a safe location” is selected because such goal has highest priority since it receives a positive influence of Norm 1.

### 5.7 Selecting Plans (SP)

The SP function is the one responsible for selecting the plan with highest priority. Let’s consider that the desires of the agents in our example with highest priority is “Evacuating the stranded workers to a safe location”, that the agent has the intention to fulfil Norm 3, and that the priority of plans that uses helicopters has decrease due to the execution of ADP function. When SP function is executed it selects the plan with highest priority that tries to rescue the NGO workers and that will not use helicopters to do so.

## 6 Related Work

We have summarized the related work in three groups: **(Norms Specification and Implementation)** Works in this group, such as [4] and [12], focus on the specification and operationalization of the norms and not on the implementation of the normative agents. Such works typically contribute with the formalization of norms and with engines that specify and explicitly manage the states of a norm. **(BDI Agents Implementation)** Although there are several of approaches to build BDI agents, for example [1] and [5], none provides support to implement normative agents. **(Normative Agents Implementation)** In this case, there are works, such as: (i) [6] that proposes an architecture to build norm-driven agents whose main purpose is the fulfilment of norms and not the achievement of their goals. In contrast, our agents are desire-driven entities that take into account the norms but are not driven by them; (ii) [3] presents concepts, and their relations, that are necessary for modelling autonomous agents in an environment that is governed by some (social) norms. Although such approach considers that the selection of desires and plans should be based on their priorities and that such priorities can be influenced by norms, it does not present a complete strategy with a set of verification in the norm review process, and strategies to evaluate, identify and solve conflicts between norms such as our work does; and (iii) [8] where the authors provide a technique to extend BDI agent languages by enabling them to enact behaviour modification at runtime in response to newly accepted norms. However, they do not answer the following questions: Why does an agent adopt a norm? How does an agent evaluate the positive and negative effects of these norms on its desires? and How does an agent detect and solve conflicts between the norms?

## 7 Conclusion and Future Work

This paper proposes an implementation to the Jason platform to build goal-oriented agents that can reason about norms. The implementation helps agents *(i)* on checking if they should adopt or not the norm; *(ii)* on evaluating the effects of the fulfilment or violation of the norm on their desires/intentions; *(iii)* on identifying and solving conflicts among norms selected to be fulfilled and among the ones selected to be violated; and *(iv)* on selecting desires and plans according to their choice of fulfilling or not a norm.

The applicability of such implementation can be verified by using the scenario presented in Section 4, where agents are responsible to plan the evacuation of people that are in hazardous location. The agents, built according to the proposed implementation, were able to reasoning about the norms they would like to fulfil, to solve conflicts among those norms, and to select plans following their intention of fulfilling or violating the norms.

We are in the process of defining an experimental study in order complete the evaluation of our approach. It is also our aim to study others BDI architectures and platforms with the aim to investigate the possibility to extend then to build BDI normative agents.

## Acknowledgments

This work has been partially supported by CNPq 557.128/2009-9 and FAPERJ E-26/170028/2008. It is related to the following topics: Software technologies for web applications - A Multi-Agent Systems Approach for Developing Autonomic Web Applications - G2. Define a framework to support the development of autonomic Web applications.

## References

1. R. Bordini, J. Hubner, and M. Wooldridge. *Programming Multi-agent Systems in AgentSpeak Using Jason*. WileyBlackwell, 2007.
2. Chris Burnett, Daniele Masato, Mairi McCallum, Timothy Norman, Joseph Giampapa, Martin Kollingbaum, and Katia Sycara. Agent support for mission planning under policy constraints. In *Proc. of 2nd Int. Technology Alliance*, 2008.
3. F. Dignum. Autonomous agents and social norms. In *In Proc. of the Workshop on Norms, Obligations and Conventions*, 1996.
4. A. García-Camino, J. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. Norm-oriented programming of electronic institutions. In *In of the Fifth International Join Conference on Autonomous Agents and Multiagent Systems (AAMAS06)*, 2006.
5. Jadex home page.

6. Martin Kollingbaum. *Norm-Governed Practical Reasoning Agents*. PhD thesis, University of Aberdeen., 2005.
7. Fabiola Lopez-Lopez. *Social Power and Norms: Impact on agent behavior*. PhD thesis, University of Southampton, 2003.
8. F. Meneguzzi and M. Luck. Norm-based behaviour modification in bdi agents. In *In Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2009.
9. Nir Oren, Michael Luck, and Timothy Norman. Argumentation for normative reasoning. In *In Proc. Symp. Behaviour Regulation in Multi-Agent Systems*, pages 55–60, 2008.
10. A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world*. Springer-Verlag New York, Inc., Rao96 1996.
11. A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. In *In Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1991.
12. Viviane Silva. From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. *Autonomous Agents and Multi-Agent Systems*, pages 113–155, 2008.