

Spatially Distributed Normative Objects

Fabio Y. Okuyama¹ and Rafael H. Bordini² and Antônio Carlos da Rocha Costa³

Abstract. Organisational structures for multi-agent systems are usually defined independently of any spatial and temporal structure. Therefore, when the multi-agent system is situated in a spatial environment, there is usually a conceptual gap between the definition of the system’s organisational structures and the definition of the environment. Spatially distributing the normative information over the environment is a natural way to simplify the definition of organisational structures and the development of large-scale multi-agent systems. By distributing the normative information in different spatial locations, we aim to allow agents to access the relevant information needed in each context. In this paper, we extend our previous work on a language for modelling multi-agent environments in order to allow for the definition of spatially distributed norms. In particular, we introduce the notion of *normative objects* and the means to distribute and handle such objects in a shared environment.

1 Introduction

The environment is an important part of Multi-Agent Systems (MAS), specially for systems of situated agents. Multi-agent systems are usually designed as a set of agents, the environment where they interact, social structures, and the possible interactions among these components. In previous works, we introduced a language that allows MAS designers to describe, at high level, environments for situated multi-agent systems [9, 1]. The language is called ELMS, and was created to be part of a platform for the development of (social) simulations based on multi-agent systems. In this paper, we present extensions which allow the distribution of normative information over an environment.

In particular, we introduce here the notion of spatially distributed *normative objects*, which in our view will facilitate the modelling of various real-world situation, particularly for simulation, but more generally for coordination of large-scale multi-agent systems.

To understand the notion of normative object, consider the posters one typically sees in public places (such as libraries or bars) saying “Please be quiet” or “No smoking in this area”. Human societies often resort to this mechanism for decentralising the burden of regulating social behaviour; people then adopt such norms whenever they have visual access to such posters. This should be equally efficient for computational systems because it avoids the need for providing a complete, exhaustive representation of all social norms in a single public structure, known to all agents, as it is usually the case in approaches to agent organisations.

Another extensions we are introducing to our environment description language is the notion of normative places, which are zones

where the normative objects are relevant. As an example, consider a research group where there are agents with the role *principal researcher* whose main objective is to supervise the research of agents playing the *research student* role; such research can be conducted both at the laboratory and at the library. The interactions at the laboratory are to be outlined in the spatial scene of the laboratory space. The information about how to behave in a library is defined in the library spatial scene, where all of them will also assume the role of *library users*. Normative information relevant for each such site (and each place within each site) can be made accessible to the agents with the help of normative objects.

In summary, the extensions we introduce here support norms and leaves the necessary room for the inclusion of group structures that are spatially situated within a (simulated) physical environment. This is done using two means: first, environment objects can now contain normative information; and second, the introduction of a special form of conditional norm, where an explicit condition on the existence of a normative object appears: ‘If an object with normative content \mathcal{N} is perceived/sensed in the environment, then reason about following norm \mathcal{N} ; otherwise, it is not required to follow that norm’. Also, normative objects may be directed toward a specific role in a given organisation. We can thus model things such as a sign saying that students are not allowed beyond the library desk (while members of staff are).

In the next section, we briefly present our platform and the various component languages we use to model multi-agent systems. In Section 3, we briefly review how an environment should be modelled using our approach. In Sections 4 and 5, we present the normative extensions that we introduce in this paper. We then illustrate our approach with an example in Section 6; the example is based on the scenario presented in [4]. We discuss related work in Section 7, then conclude the paper.

2 The MAS-SOC Platform

One of the main goals of the MAS-SOC simulation platform (**M**ulti-**A**gent **S**imulations for the **S**ocial **S**ciences) is to provide a framework for the creation of agent-based simulations which do not require too much experience in programming from users, yet allowing the use of state-of-the-art agent technologies. In particular, it should allow for the design and implementation of simulations with *cognitive* agents.

In our approach, an agent’s individual reasoning is specified in an extended version of AgentSpeak [11], as interpreted by *Jason*, an *open source* agent platform⁴ based on Java [2]. The extensions allow, among other things, for speech-act based agent communication, and there is ongoing work to allow for ontologies as part of an AgentSpeak agent’s belief base.

¹ Universidade Federal do Rio Grande do Sul, Brazil, email: okuyama@inf.ufrgs.br

² University of Durham, United Kingdom, email: R.Bordini@durham.ac.uk

³ Universidade Católica de Pelotas, Brazil, email: rocha@atlas.ucpel.tche.br

⁴ Available at <http://jason.sf.net>.

The environments where agents are situated are specified in ELMS, a language we have designed for the description of multi-agent environments[9]. For more details on MAS-SOC, refer to [1]. We here concentrate on the ELMS extensions to describe basic organisational structures and social norms, and to relate an organisational structure and the relevant normative aspects to the spatial structures defined within the physical environment.

3 Modelling Physical Environments

As presented in [9], we developed an language to describe environments and the means to execute the simulated environment. Agents in a multi-agent system interact with the environment where they are situated and interact with each other (possibly through the shared environment). Therefore, the environment has an important role in a multi-agent system, whether the environment is the Internet, the real world, or some simulated environment.

We understand as environment modelling, the modelling of external aspects that an agent needs as input to its reasoning and for deciding on its course of action. Further, it is necessary to model explicitly the physical actions and perception capabilities that the agents are allowed in a given environment.

The language is called ELMS (**E**nvironment **D**escription **L**anguage for **M**ulti-**A**gent **S**imulation). Below we briefly review how a physical environment is described using this language.

To define an environment using ELMS, the following classes of constructs can be used:

- **Agent Body:** an agent’s characteristics, that are perceptible to other agents. The “bodies” are defined by a set of properties that characterise it and are perceptible to other agents. Such properties are represented as *string*, *integer*, *float*, and *boolean* values. Each “body” is associated with a set of actions and the properties that the agent can perceive.
- **Agent Sensorial Capabilities (Perception):** the environment properties that will be perceptible to each agent at a given time, and under given specific circumstances.
- **Agent Effective Capacities (Actions):** the actions that an agent is able to perform in order to change the current state of the environment. These actions are defined as assignments of values to the attributes of environments⁵. The production (instantiation) of previously defined resources (i.e., objects), and the consumption (deletion) of existing instances may also be part of an action description.
- **Physical Environment Objects (Resources):** the objects/resources that are present in the environment. Although objects and resources can have some conceptual differences, they are represented by same structure in ELMS. Agents interact with objects through the actions over the environment. Object structures are defined by a set of properties that are relevant to the modelling and may be perceived by an agent. In the same way as the “bodies” of the agents, the resources are represented by *string*, *integer*, *float*, and *boolean* values. Each object can also be associated with a set of reactions that may happen as consequence of agent actions.
- **Reactions:** the objects can “react”, on specific circumstances, in order to respond to actions performed by the agents in the environment. It is defined by a set o values assignments, creation of previously defined object instances, and the deletion of existing object instances.

⁵ Note that agent bodies are also properties of the environment.

- **Spatial Representation (Grid):** the space is (optionally) divided into cells forming a grid that represents the spacial dimension of the environment. The Grid can be defined with 2 or 3 dimensions. Each cell can have reactions (as the resources) associated to it. Although the specified set of reactions apply to all of the cells, this does not mean that all cells will behave equally, since they may have a different contexts (i.e., each cell has independent attributes, thus having different contents and, clearly, different positions, which can all affect the particular reactions).

Notes on Environment Descriptions

- **Perceptions:** agents do not normally have complete access to the environment. Perception of the environment will not normally give complete and accurate information about the whole environment and the agents in it. However, since this is not restricted by the ELMS model, designers can choose to create fully accessible environments.
- **Actions:** it helps maintaining the coherence of the environment if actions are “atomic”. This creates the possibility of another agent or the environment interfere on the realisation of a chain of actions. The action chaining or planning is meant to be part of the “mind” of the agent, a whole plan should not be implemented as an action available to agent in the environment. Also, *atomic actions* allow a bigger set of possible action chaining.
- **Reactions:** all reactions triggered by some change in the environment are executed in a single simulation cycle. This is different from actions, as each agent can execute only one action per cycle.

Additionally to the constructs mentioned above, the following operational constructs are used in our approach to model the (physical) environment.

- **Constructors:** Each agent and resource may need to be initialised at the moment of its instantiation. This is defined by a list of value assignment to its attributes.
- **Observables:** A list of environment properties whose values are displayed/logged; these are the specific properties of a simulation that the user wants to observe/analyse.

The simulation of the environment itself is done by a process that controls the access and changes made to the data structure that represents the environment; the process is called the *environment controller*. The data structure that represents the environment is generated by the ELMS interpreter for a specification in ELMS given as input. In each simulation cycle, the environment controller sends to all agents currently taking part in the simulation the percepts to which they have access (as specified in ELMS). Recall that ELMS environments are designed for cognitive agents, so perception is transmitted in messages as a list of ground logical facts. After sending perception, the process waits for the actions that the agents have chosen to perform in that simulation cycle and then execute the actions, changing the environment data structures.

4 Normative Places

As described in previous sections, we have developed a language to describe environments for situated multi-agent systems. The description, based on the concepts of agent bodies, objects, and an optional grid, did not offer the means to define the notion of a “place”, i.e., that a set of cells would be related to a concept of a place with similarities: places where activities are done or places where groups or agent

settings are related; we refer to them as *Normative Places*. This is effectively used to represent the physical space where an organisation takes place; that is, the spatial scope of a particular organisation, and consequently the norms related to the activities of such organisation.

A *Normative Place* is defined simply by its name and the set of cells that are part of it. Each *normative place* is a set of cells with a label, that may have intersection with other sets, or even be contained by another. For example a school may have a large set of cells where some cells refer to a normative place “classroom”, and another to its “library”. The *normative place* definition would allow for the definition of the spatial location where certain norms are valid and relevant, as it will be seen in the next section. As future work, a *normative place* is intended to be also associated to group structures, creating a connection between the organisational structures and the physical environment. We plan to make possible the association of any existing approach to agent organisations, such as MOISE⁺ [8], OperA/OMNI [12], GAIA [14], or approaches based on “electronic institutions” [5, 6], to each normative place.

In order to ease the definition of repetitive normative place structures, classes can be defined then “instantiated” in specific positions of the grid. Examples of such definitions are as follows.

```
<PLACE NAME="library">
  <CELL X="0" Y="0"/>
  <CELL X="0" Y="1"/>
</PLACE>

<PLACE NAME="classroom">
  <CELL X="2" Y="0">
</PLACE>
```

5 Normative Objects

Typically, environments will have some objects aimed at informing agents about norms, give some advice, or warn about potential dangers. For example, a poster fixed on a wall of a library asking for “silence” is an object of the environment, but also informs about a norm that should be respected within that space. Another example are traffic signs, which give advice about directions or regulate priorities in crossings. The existence of such signs implies the existence of a regulating code to be followed in such places.

In this extended version of ELMS, there are special types of objects that contain normative information, which we refer to as *normative objects*. Those objects are “readable” by agents under specific individual conditions. For example, an agent can read a specific rule if it has a specific ability to perceive that type of object. In the most typical case, the condition is simply being physically close to the object.

Such objects can be defined before the simulation starts, or can be created dynamically during the simulation. Each object can be placed in a collection of cells of the spatial grid of the environment. Such cells represent the *normative place* where the content of the normative object can be accessed and is relevant. If such collection of cells is not given, the normative object will only be perceived by agents under specific conditions. The conditions under which the normative objects can be perceived are defined by the simulation designer using the usual ELMS constructs for defining conditions.

The normative information in a normative object is “read” by an agent through its perceptive ability. It contains the norm itself and meta-information (e.g., which agent or institution created the norm). The normative objects can be defined before the simulation starts in a *norms definition file* or during the simulation, by the definition of the following properties:

- **Type:** the type of the normative information contained in the object; it determines the level of importance (e.g. a warning, an obligation, a direction);
- **Issued by:** where the power underlying the norm comes from (e.g., an agent, a group, an institution).
- **Norm:** a string that represents the normative information; this should be in the format of AgentSpeak predicates in the case of MAS-SOC environments, or whatever format the agents will be able to understand.
- **Placement:** the label of a place, or $\langle x, y, z \rangle$ coordinates (which may be omitted). If omitted, the object can be accessed from anywhere, but normally at conditions determined by the designer; see the next item. This also determines the space where the normative information applies.
- **Condition:** conditions under which the normative information can be perceived. The conditions can be associated with groups, roles, abilities, and current positions of agents and objects.
- **Id:** string identification for eventual deletion/editing of the normative object.

We now briefly describe how the agents will receive normative information from normative objects. Whenever the agent position is such that access to the normative object is granted, and the **Condition** is satisfied, the agent will get a perception of the form:

[kind]([Issued by],[NORM],[PLACE])

Ex: rule(family, obligation(child,play(TOY),tidy(TOY)),home)

The example above can be read as: “There is rule in group *family*, with application at *home*, that says: if the action *play(TOY)* is done by an agent of role *child*, then it is an obligation of that agent to do *tidy(TOY)* as well”. A rule like that would not normally be posted on a sign at a home, but it illustrates the more general idea of normative objects as norms that apply within given environmental locations.

It is important to note that the norm-abiding behaviour is not related to the existence of a normative object. Beyond the existence of such object, it is necessary for the agent to perceive the normative object, and reason about following or not the norm stated by the normative object.

The main advantage of modelling norms in this way is the fact that the spatial context is determined. Thus, the norm can be contextualised, thus being less abstract, which makes it easier to operationalise the norm and also facilitates verification of norm compliance. For example a norm that says “Be kind to the elderly”, may be quite hard to operationalise and verify, while in a fixed spatial context such as a bus or train, that norm can be *contextualised* as “give up your seat for the elderly”, which would be much more easily interpreted and verified.

Modelling Multi-Agent Systems

Since our platform does not enforce a particular agent-oriented software engineering methodology, designers can use the one of their preference. It is possible to model a multi-agent system that will have an ELMS environment using any approach: starting from the organisations (top-down), or starting from the agent reasoning (bottom-up).

In both approaches, the modelling of the organisational structures and the agent reasoning needs fine tuning to achieve the desired results. To have a stable point on which to base the tuning-up of the agents’ reasoning or the organisational model, we have suggested the use of an explicitly defined environment description written in

the ELMS language and the concepts presented in the Section 3. The environment is an important part of an multi-agent system, and although it can be very dynamic, in regards to design it is usually the most “stable” part of the system.

Even when the environment of the multi-agent system is the “real world” and the agent is a robot with sensors and effectors, the environment modelling should play a significant role. In such situation, the robot should have a set of sensors that give a predefined set of percepts the robot will acquire when sensing the environment. Also, the robot should have a set of effectors that allow a restricted set of (parameterisable) actions. Thus, the possible sensor inputs and effectors output should be modelled first to facilitate the development of the software for the robot.

After that initial modelling, defining which are going to be agents in the simulation and the general goals of the organisations in the society, we suggest starting the implementation using the environment language. The ELMS language is meant to describe the environment, by defining its spatial representation, the types of objects present in the environment, and the types of agents. Each type of agent is defined with its sensorial and effective capacities.

For the reasons mentioned above, we suggest that the multi-agent modelling starts with the environment definition, followed by the definition of the normative places. The environment modelling is achieved by:

1. Definition of which kinds of action each type of agent is able to perform in the environment. Actions typically produce effects over objects of the environment or other agents.
2. Based on the changes that the agents’ effective capabilities are able to make in the environment and the objectives of the simulation, the size and granularity of the grid can be defined. For example, how many cells an agent can move within one action or simulation cycle, and in how many simulation cycles the agent would be able to traverse the simulated space.
3. Based on the granularity and size of the spacial environment, the sensorial capacities of the agents can be modelled, defining for example in which range an agent can detect other agents or objects.
4. Based on an agent’s sensorial capabilities and on its main goals, it should be possible to define which attributes of that agent is important to declare as accessible to other agents . For example, if agents identify each other by colours, the “agent body” should have an attribute that represent the agent’s colour.
5. The types of objects or resources present in the environment should also be modelled based on which attributes will be perceptible by the agents and which actions can affect them.
6. Finally, instances of the agent and object classes should be placed in the environment, to define its initial state.

The definition of normative places should be followed by the definition of spatially distributed norms:

1. Together with the objects instances placed on the environment, the set of normative places within the environment can also be defined.
2. By assigning labels to a set of cells, a *normative place* is created.
3. Then, based the set of activities that can be performed there, the norms that are relevant to that place can be defined.
4. Finally, the *normative objects* can be defined, defining the places where the norms can be perceived.

Using the environment as a basis, the agents’ reasoning capabilities can be defined so as to help agents achieve their own objectives as well as the objective of the groups in which they participate. Also,

the detailed definitions of possible organisational structures can be fine tuned, in order to have the system achieving its overall objectives. In our approach, we normally use AgentSpeak to define the practical reasoning for each agent; in particular, we use the extended version of AgentSpeak as interpreted by *Jason*; for details, see [3].

6 Example

This is an example showing how normative objects are defined using our approach. It is based on the scenario presented in [4], a scenario where the agents are placed on an environment where they may eat the food they find, challenge other agents for their food, or move in search of food.

In this scenario, an agent owns any food item that is near to itself (at a distance of up to 2 cells). The agents can “see” food and other agents in a radius of 1 cell, but can sense food in a radius of 2 cells. The physical space is represented by a grid of 10 × 10 cells.

The norm is basically to respect the ownership of a food, which means non-aggressive behaviour. In the original scenario the norms were valid throughout the grid, but in this example norms are valid in normative places containing normative objects.

A shortened version of the physical environment description in ELMS is given below.

```
<!DOCTYPE ENVIRONMENT SYSTEM "elms.dtd">
<ENVIRONMENT NAME = "NORMATIVE">
  <DEFGRID SIZEX="10" SIZEY = "10"/>

  <RESOURCE NAME="food">
    <STRING owner = "none">
  </RESOURCE>

  <AGENT_BODY NAME="agent">
    <INTEGER NAME = "id"> "SELF" </INTEGER>
    <PERCEPTIONS>
      <ITEM NAME = "vision"/>
      <ITEM NAME = "sense_food">
    </PERCEPTIONS>
    <ACTIONS>
      <ITEM NAME = "walk"/>
      <ITEM NAME = "attack"/>
      <ITEM NAME = "eat"/>
    </ACTIONS>
  </AGENT_BODY>

  <PERCEPTION NAME="vision">
    <CELL_ATT ATTRIBUTE="CONTENTS" ABSOLUTE="TRUE">
      <X> +0</X>
      <Y> +0</Y>
    </CELL_ATT>
    <CELL_ATT ATTRIBUTE="CONTENTS" ABSOLUTE="TRUE">
      <X> +1</X>
      <Y> +0</Y>
    </CELL_ATT>
    <!-- summarised-->
  </PERCEPTION>

  <PERCEPTION NAME="sense_food">
    <!-- summarised-->
  </PERCEPTION>

  <ACTION NAME="eat">
    <PARAMETER NAME = "FOOD_ID" TYPE="INTEGER" />
    <!-- summarised-->
  </ACTION>

  <ACTION NAME="walk">
    <!-- summarised-->
  </ACTION>

  <ACTION NAME="attack">
    <!-- summarised-->
  </ACTION>

  <INITIALIZATION>
    <!-- instantiation and placement of
      food and agents -->
  </INITIALIZATION>
</ENVIRONMENT>
```

In the code sample above, the grid size is defined, then food is defined as an environment resource, then a generic type of agent body is defined. The agent body is defined as having two types of perception (vision and food sensing) and is able to perform three types of actions (walk, attack, and eat). The vision perception allows the agent to perceive the contents of the current cell and the 4 neighbouring cells, while sense_food allows it to perceive food in a radius of 2 cells.

For this example, the grid is partitioned in four normative places of equals size, and the normative objects are defined and placed only in the upper-left quadrant, as shown in the code sample below:

```
<PLACE NAME="first">
  <CELL X="0" Y="0" /><CELL X="1" Y="0" />
  <!-- summarised-->
  <CELL X="3" Y="4" /><CELL X="4" Y="4" />
</PLACE>

<PLACE NAME="second">
  <CELL X="5" Y="0" /><CELL X="6" Y="0" />
  <!-- summarised-->
  <CELL X="8" Y="4" /><CELL X="9" Y="4" />
</PLACE>

<NORM_OBJ ID="norm1" TYPE="norm" PLACE = "first">
  <NORM>ought (~attack(AGENT))</NORM>
</NORM_OBJ>

<NORM_OBJ ID="norm2" TYPE="norm" PLACE = "first">
  <NORM>ought (~eat(owned_food))</NORM>
</NORM_OBJ>
```

The norms defined above are very simple, and are given simply to illustrate how they can be modelled in our approach. For example, norm1 says that an agent ought not to attack another agent (i.e., it is a prohibition), while norm2 says that the agent ought not to eat the food that has another owner.

7 Related Work

The notion of artifacts [13] and coordination artifacts [10] resembles our notion of *normative objects*. As defined in [10], coordination artifacts are abstractions meant to improve the automation of coordination activities, being the building blocks to create effective shared collaborative working environments. They are defined as runtime abstractions that encapsulate and provide a coordination service to the agents. Artifacts [13] were presented as a generalisation of coordination artifacts. The artifacts are an abstraction to represent tools, services, objects and entities on a multi-agent environment. As building blocks for environment modelling, artifacts encapsulate the features of the environment as services to be used by the agents.

In our work, rather than having a general notion of objects that by their (physical) properties facilitate coordination, *normative objects* are objects used specifically to store *symbolic* information that can be interpreted by agents so that they can become aware of norms that should be followed within a well-defined location (of course agents, being autonomous, will decide whether to follow the norms or not). An important difference is that *normative objects* are spatially distributed over a physical environment, with a spatial scope where they apply, and closely tied to the organisation that physically located in that space. While the objective of the coordination artifacts is to remove the burden of coordination from the agents, our work tries to simplify the way designers can guide the behaviour of each individual agent as they move around an environment where organisations are spatially located; this allows agents to adapt the way they behave in different social contexts.

In [7], the authors present the AGRE model, an extension to the previous AGR model. These latest extensions allow the definition of structures that represents the physical space. The approach defines organisational structures (groups) and the physical structures (areas) as “specialisations” of a generic “space”. The social structures are not contextualised in the space as in our work, leaving the social and physical structures quite independent. In our work it is not possible to explicitly define the social structures, even though it would be possible to define them through the norms. This is because our aim is, as we mentioned earlier, to make our environmental infrastructure compatible with existing approaches to organisational modelling. Also, note that we are only interested in cognitive multi-agent systems in particular.

8 Conclusions

In this paper we have extended our language for describing environments with the means to define normative structures that make part of an environment representation. There are currently many approaches to modelling and implementing multi-agent systems. There are the top-down approaches with focus on the organisations, and bottom-up approaches with focus on the agents. We believe that including environment modelling at the initial stages of both approaches would help the modelling and implementation of multi-agent systems. To help such modelling, we have proposed an approach with an explicit environment description which now also includes spatially distributed normative information.

It is important to notice that our work is not an approach for modelling the organisational dimension of a multi-agent system. With the definition of *normative places*, where group structures would be inserted, we intend to fill a conceptual gap between the usual ways organisations and physical environments are modelled. In future work, with the possibility of associating organisational structures of existing approaches to the normative places, we hope to contribute to a more integrated approach to designing and implementing the various aspects of multi-agent systems. Concentrating in one particular context, which is also limited to a spatial location, makes it easier for designers to define the groups and roles that operate at that context.

By distributing the normative information in the environment, it is possible to partition the environment in a functional way, thus helping the structured definition of large simulations, norms being associated only with the places where they should be followed. It is also more efficient (by taking advantage of natural distribution) to have norms spread in an environment than having a repository for the whole society, as it is usually the case.

We believe that an explicit environment description is an important part of a multi-agent system due to the fact that it is a stable point from where the agent reasoning and the organisational structures can be fine tuned so as to facilitate the development of agents and organisation that can achieve their goals. The notion of *spatially distributed normative objects* that we have introduced here can be a good solution connecting definitions of organisations and definitions of environments. Additionally, distributing the organisational/normative information can ease the modelling of large organisations.

It is interesting to note that, being conditioned on the possibility of checking the existence of a normative object, the normative reasoning required from agents that deal with normative objects is necessarily of a non-monotonic nature, and the experience of programming such reasoning in AgentSpeak is something we plan to experiment with in the future.

In future work, we plan not only to develop simulations in order to evaluate and improve our approach, but also to experiment with this form of non-monotonic reasoning in AgentSpeak. The “norm reasoning” associated to the possibility of defining normative places within the environment, each one with its own organisational purposes and sets of norms, brings many issues to be addressed in the future.

ACKNOWLEDGEMENTS

This work was partially supported by CNPq, CAPES, and FAPERGS. Rafael Bordini gratefully acknowledges the support of The Nuffield Foundation (grant number NAL/01065/G).

REFERENCES

- [1] Rafael H. Bordini, Antônio Carlos da Rocha Costa, Jomi F. Hübner, Álvaro F. Moreira, Fabio Y. Okuyama, and Renata Vieira, 'MAS-SOC: a social simulation platform based on agent-oriented programming', *Journal of Artificial Societies and Social Simulation*, **8**(3), (2005).
- [2] Rafael H. Bordini, Jomi F. Hübner, et al., *Jason: A Java-based agentSpeak interpreter used with saci for multi-agent distribution over the net*, manual, release version 0.7 edn., Aug 2005. <http://jason.sourceforge.net/>.
- [3] Rafael H. Bordini, Jomi F. Hübner, and Renata Vieira, 'Jason and the Golden Fleece of agent-oriented programming', in *Multi-Agent Programming: Languages, Platforms and Applications*, eds., Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, chapter 1, Springer-Verlag, (2005).
- [4] C. Castelfranchi, R. Conte, and M. Paolucci, 'Normative reputation and the costs of compliance', *Journal of Artificial Societies and Social Simulation*, **1**(3), (1998). <<http://www.soc.surrey.ac.uk/JASSS/1/3/3.html>>.
- [5] Marc Esteva, David de la Cruz, and Carles Sierra, 'Islander: an electronic institutions editor.', in *AAMAS*, pp. 1045–1052. ACM, (2002).
- [6] Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Lluís Arcos, 'Ameli: An agent-based middleware for electronic institutions.', in *AAMAS*, pp. 236–243. IEEE Computer Society, (2004).
- [7] Jacques Ferber, Fabien Michel, and José-Antonio Báez-Barranco, 'Agre: Integrating environments with organizations.', in *E4MAS*, pp. 48–56, (2004).
- [8] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier, 'MOISE⁺: Towards a structural, functional, and deontic model for MAS organization', in *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2002)*, Bologna, Italy, (2002).
- [9] Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa, 'ELMS: An environment description language for multi-agent simulations', in *Proceedings of the First International Workshop on Environments for Multiagent Systems (E4MAS)*, held with *AAMAS-04, 19th of July*, eds., Danny Weyns, H. van Dyke Parunak, and Fabien Michel, number 3374 in Lecture Notes In Artificial Intelligence, pp. 91–108, Berlin, (2005). Springer-Verlag.
- [10] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini, 'Coordination artifacts: Environment-based coordination for intelligent agents', in *AAMAS'04*, (2004).
- [11] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, 22–25 January, Eindhoven, The Netherlands, eds., Walter Van de Velde and John Perram, number 1038 in Lecture Notes in Artificial Intelligence, pp. 42–55, London, (1996). Springer-Verlag.
- [12] Javier Vázquez-Salceda, Virginia Dignum, and Frank Dignum, 'Organizing multiagent systems.', *Autonomous Agents and Multi-Agent Systems*, **11**(3), 307–360, (2005).
- [13] Mirko Viroli, Andrea Omicini, and Alessandro Ricci, 'Engineering MAS environment with artifacts', in *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, eds., Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, pp. 62–77, AAMAS 2005, Utrecht, The Netherlands, (26 July 2005).
- [14] Michael Wooldridge, Nicholas R. Jennings, and David Kinny, 'The GAIA methodology for agent-oriented analysis and design', *Autonomous Agents and Multi-Agent Systems*, **3**(3), 285–312, (2000).