Spatially Distributed Normative Infrastructure

Fabio Y. Okuyama[1], Rafael H. Bordini[2], and Antônio Carlos da Rocha Costa[3]

[1] Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre–RS, Brazil
okuyama@inf.ufrgs.br

[2] Department of Computer Science – University of Durham
Durham DH1 3LE, U.K.
R.Bordini@durham.ac.uk

[3] Escola de Informática – Universidade Católica de Pelotas (UCPel)
Pelotas–RS, Brazil
rocha@atlas.ucpel.tche.br

Abstract. In previous works we have presented a model to describe and simulate environment for situated multi-agent systems, that we called ELMS. Here, we present an extensions to our model that provide means to have normative information distributed in the environment. Organisational structures for multi-agent systems are usually defined independently of any spatial or 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 allow agents to directly access the relevant information needed in each environmental context. The extensions to our model for multi-agent environments, allows the definition of spatially distributed norms 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), even more 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 presented a language that allows MAS designers to describe, at a high level, environments for situated multi-agent systems [12, 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 that complements the environment description with structures that by allowing the distribution of normative information over the environment supporting a connection among the environment and the organisational structures.

In particular, we present here a set of concepts for a spatially distributed infrastructure formed by *normative objects*, *normative places* and *norm supervisors*. This infrastructure in our view will facilitate the modeling of various real-world situation, particularly for simulation, but more generally also 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 having all the norms hard-wired in the agents or the need to 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.

The *normative places* are zones were the normative objects are pertinent. As an example, consider a research group where there are *researcher* agents and its main objective is to do its research, what can be done at the laboratory and at the library. The interactions at the laboratory among researchers, staff and with the environment 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 location, where the researches will also assume the role of *library user*. Normative information relevant for each such site (and each place at each site) can be posted to the agents with the help of normative objects. The *norm supervisor* agents, are a special class of agents that monitors the compliance of norms in the MAS. What differs in *norm supervisors* agents is that they are enabled to receive normative information that are not specifically relevant to them. They may be system agents or plain agents with such functionality enabled.

In summary, the extensions we introduce here support situated 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, *normative objects*, which are objects that can contain normative information; and second, a normative principle for *situated norms*, conceived as a special form of conditional rule, where an explicit condition on an agent's perception of a normative object appears: 'When playing the relevant role and being physically situated within the confines referred by a situated norm $\mathcal{N}$ expressed in a normative object previously perceived, the agent is required to reason about following norm $\mathcal{N}$; otherwise, it is excused from reason about it'. Also, normative objects may be directed towards 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 Section 4, we present the normative extensions that we introduce in this paper. We then illustrate our approach with an example in Section 5; the example is based on the scenario presented in [4]. We discuss related work in Section 6, 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 **SOC**ial Sciences) 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 [14], as interpreted by ***Jason*** [2], an Open Source agent platform[4] based on JAVA. 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.

The environments where agents are situated are specified in ELMS, a language we have designed for the description of multi-agent environments [12]. 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 [12], 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. In a multi-agent scenario, how one agent percepts another is an important issue. Thus, modelling explicitly the agent "body" (or avatar) should also be included on environment modelling. 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 Description **L**anguage for **M**ulti-Agent **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:** the agent's characteristics that are perceptible to other agents. Agent "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 that the agent is allowed to perform and of environment properties that the agent can perceive.

[4] Available at `http://jason.sourceforge.net`.

– **Agent Sensorial Capabilities (Perception):** specifies what kind of the environment properties that will be perceptible to each agent that have a "body" with such capacity. It is defined as the environmental properties that will be sent to the agent and the specific circumstances where they are possible (e.g. an agent may be able to see in a radius of 2 cells only if there are nothing obstructing its vision).
– **Agent Effective Capacities (Action):** specifies what kind of environment changes that is possible to an agent that have a "body" with such capacity. These changes are defined as assignments of values to the attributes of environments[5]. The production (instantiation) of previously defined resources (objects), and the consumption (deletion) of existing instances may also be part of an action description. Also, the conditions under such action may be successfully[6] performed should be specified, for example a robot may walk only on a certain type of terrain, i.e. the action will fail if the terrain type is wrong.
– **Physical Environment Objects and Resources:** the objects and 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 an agent's actions.
– **Object Reactions:** the objects can "react", under specific circumstances, in order to respond to actions performed by the agents in the environment. Such reactions are given as the assignment of values to properties, the creation of previously defined object instances, and the deletion of existing object instances.
– **Space Structure (Grid):** the space is (optionally) divided into cells forming a grid that represents the spatial structure of the environment. When a grid is used, it can be defined in 2 or 3 dimensions. As for resources, each cell can have reactions 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 be in different contexts (i.e., each cell has independent attributes, thus having different contents and, clearly, different positions, which can all affect the particular reactions).

3.1 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 other agents in it. However, since such restriction is not imposed by the ELMS model itself, designers can choose to create fully accessible environments if this is appropriate for a particular application.

[5] Note that agent bodies are also properties of the environment.
[6] In the current implementation of the model it is only possible to perform successful actions, otherwise the action fails completely, actions with partial success may be available in future versions.

– **Actions:** it helps improve the coherence of the environment if agent actions are defined as simple "atomic" actions. This creates, of course, the possibility of another agent or the environment interfere on the realisation of an agent's plan (chain of actions). However, as reasoning about action choices is meant to be part of the agents' "mind", a whole course of action should not be defined as a single, complex action made available to agents by the environment definition. Also, "simple atomic actions" allow a wider range of possible action interleavings.
– **Reactions:** all object reactions triggered by some change in the environment are executed in a single simulation cycle. This is different from agent 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 (simulated) 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 initial value assignment to its attributes.
– **Observables:** A list of environment properties whose values are to be 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 (in fact, only such process can access the data structure); 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 action over the environment, what means to perform the changes specified in the actions on the environment data structures. Finally, the environment sends the updated perceptions to the agents, starting a new cycle.

4 Normative Infrastructure

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, that we call *normative objects*, implies the existence of a regulating code in such context, that we call *situated norms*.

In the examples above, the norms are only meant to be followed within certain boundaries of space or time and lose their effect completely if those space and time restrictions are not met, which is the initial motivation for situated norms. Another

important advantage of modelling some norms as situated norms is the fact that the spatial context where the norm is to be followed is immediately determined. Thus, the norm can be "pre-compiled" to its situated form, making it easier for the agents to operationalise the norm, and also facilitating the verification of norm compliance.

In this section we present the extensions to ELMS, these extensions are meant to provide an infrastructure in order to allow the distribution of normative information on the environment. Such infrastructure intends to be a connection point between the environment and organisational structures, improving significantly the possibilities of our simulation platform.

4.1 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 $\mathcal{M}$OISE$^+$ [11], OperA/OMNI [16], GAIA [18], or approaches based on "electronic institutions" [7, 8], 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.

```
<NORMATIVE-PLACE-TYPE NAME="library"/>

<NORMATIVE-PLACE-TYPE NAME="classroom"/>

<PLACE NAME="lib1" NORMATIVE-PLACE-TYPE="library">
     <CELL X="0" Y="0"/>
     <CELL X="0" Y="1"/>
</PLACE>

<PLACE NAME="cr1" NORMATIVE-PLACE-TYPE="classroom">
     <CELL X="2" Y="0">
</PLACE>
```

4.2 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 targeted agents will be able to understand.
- **Placement:** the set of normative spaces where the normative information applies. If omitted, the object is assumed to be accessible from anywhere, but normally under 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 physical placement and orientation of agents and objects.
- **Id:** identification string for eventual deletion/edition 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 accessible, and the **Condition** is satisfied, the agent will receive perception of the form:

rule([PLACE],[GROUP],[ISSUED BY],[NORM])
Ex: rule(home, family, parents, obligation(child,play(TOY),tidy(TOY)))

The example above can be read as: "This is rule in group *family*, issued by the *parents*, with application at the normative place *home* (see below), 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 in a family home, but it illustrates the more general idea of situated norms as norms that apply within given environmental locations.

It is important to remark 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 autonomous agents will also reason about whether to follow or not the norm stated by the normative object.

4.3 Pre-compilation of Norms

The *normative objects* are not meant to be only means to spread general norms. The norms informed through the normative objects are supposed to be contextualised or *pre-compiled*[7] with the information of the determined *normative place* where it is pertinent.

Since the spatial context of the norm is limited and determined by the normative places, a generic norm can be *pre-compiled* with such information, in order to become less abstract. This process is meant to make easier to operationalise the norm, since the norm is "ready to use" and it is in the spatial context where it is pertinent. Other advantages of having less abstract norms are that the verification of norm compliance is facilitated and the reduction of misinterpretations of the norms, what could happen with abstract non-contextualised norms.

For example a norm that says "Be kind to the elderly", may be quite hard to operationalise and verify, in general. However, in a fixed spatial context such as a bus or train, with the norm contextualised as "Give up your seat for the elderly", or in a street crossing, with the norm contextualised as "Help elderly people to cross the street", the norm would be much more easily interpreted by the agents, and verified by any norm compliance checking mechanism.

4.4 Norm Supervisors

Since the agents are free to reason about abiding or not a norm stated in a normative object, there is also a need to monitor the behaviour of those agents.

The supervisor agent may be an agent of the system, designed to verify the compliance of the norms or it may be a common agent whose interests require that all the agents follow the norms. In both cases, the agent just need to be enabled to receive some extra informations about the norms, the actions being performed and environmental state changing.

Since the norm and the possible violations are contextualised in a specific place in a normative place, it is much more easier to define the possible violations and the norms

[7] Task done by the simulation designer.

that are pertinent to such place. By the use of simple rules the supervisor agent verifies the compliance of norms, and then according to the requirements of the system, it may interrupt a course of action, punish or simply report infractions of the norm.

It is important to notice that, the norm supervisors are not meant to avoid infractions. The norm supervisors are agents that have access to extra information in order to be able to verify the compliance of norms. The simulation designer may enable such capacity in a agent to help it achieving its goal or use such informations to monitor the simulation or as an input for a reputation system.

According to [6] an agent may be motivated to verify the compliance of the norms by other agents in order to assure that the costs of norm adherence is being paid by the other agents too. A norm abiding agent, will want that all the others addressees of the norms follow it too, otherwise the norm adhering behaviour may become a kind of competitive disadvantage. In [6], the authors refer to agents with such behaviour as *norm defenders*, it is also said that an agent complying with norms is likely to become a *norm defender*.

4.5 Environment-Agent Cycle

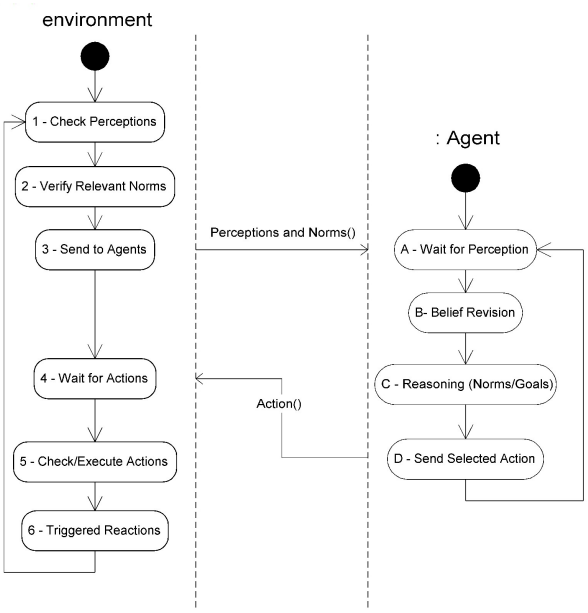

Fig. 1. Environment-Agent Cycle

In this section, we present the general lines of how the environment is simulated. The Figure 1 shows on the left side some activities states for the environment simula-

tion, on the right side a simplified model of activities states of an agent, with the main communications at the center. The environment states are described bellow:

1. The environment checks the class of the agent and which perceptions are enabled for such class of the agent. Then, it verifies which perceptions fulfills the conditions specified on the perception specification and stores the specified environment properties to be sent to the agent.
2. The normative objects that are in the same normative place are checked in order to verify if they contain normative information relevant to the agent. If the conditions are fulfilled the norms are also stored to be sent.
3. The environment sends the perceptions and normative information to the agent.
4. The environment waits for the action selected by the agent.
5. It checks if the selected action is enabled to the agent and if it fulfills the conditions specified on the action specification. If so, environment properties are changed as speficied on the action definition.
6. The environment verifies which reactions of the objects were triggered by the current state of environment properties. The triggered reactions are performed over the environment data. And then, the cycle starts over going back to the step 1.

A general outline of a *protocol* that the agent should follow is described bellow:

A The agent waits for the reception of the perceptions and normative information from the environment.
B The agent may compare the perceptions received with its own world representation in order to detect changes.
C The agent will select its action, based on its own objectives and the norms that it has perceived. Such decision involves the balance of a large set of factors, from where we may cite:
   – beliefs about the environment;
   – goals or states-of-world that the agent want to become true;
   – determine which goals from the set of are possible;
   – selection of which goal to follow;
   – selection of which norms to follow or not;
   – solve conflicts among goals and norms;
D The agent sends the selected perception.

If the agent is an supervisor agent, the main differences will be that it will be enabled to receive normative information that may be not relevant for itself and extra information about other agents. Using such information the supervisor agent may verify the compliance of norms. The actions that the supervisor agent may do in a case of non-compliance of norms is

Modelling Environments using ELMS

As the MAS-SOC platform does not enforce a particular agent-oriented software engineering methodology, designers can use the one they prefer. It is possible to model

a multi-agent system that will have an ELMS environment using any approach: starting from the system organisation (top-down), or starting from the agent interactions (bottom-up).

In both approaches, the modelling of the organisational structures and the agents' reasoning need 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 suggest 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.

Based on these observations, we suggest that the multi-agent system 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 determined. 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 spatial environment, the sensorial capabilities 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 typical activities, 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's role by the colour of their uniform, the "agent body" should have an attribute that represent the colour of the agent's uniform.
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, determining its initial state.

The definition of the environment should be followed by the definition of normative places and then by the definition of the spatially distributed normative objects, as follows:

1. Together with the objects instances placed in the environment, the types of normative places within the environment can also be defined.
2. By instantiating normative places into sets of cells, *normative places* are created.
3. Then, based the set of activities that can possibly be performed in each type of normative place, the norms that are relevant to that type of place can be defined.
4. Finally, the types of *normative objects* can be defined and instantiated in the normative places, defining the locations where situated norms can be perceived.

Using the environment as a basis, the agents' reasoning capabilities can then be defined so as to help agents achieve their 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 MAS-SOC, we 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].

5 Example

Below we give an example showing how normative objects are defined using our approach. It is based on the scenario presented in [4], a scenario in which 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 norms used in that scenario essentially concern the respect for the ownership of a food item, which means they prescribe non-aggressive behaviour. In the original scenario, the norms were valid throughout the grid, but in this example norms are valid only within normative places, as indicated by normative objects.

A shortened version of the physical environment description given below. The actual environment description of environments in ELMS are done in XML, but in order to optimise the space and improve readability, we have adopted a pseudocode just to show the main points of the environment.

```
environment.name:="NORMATIVE";
environment.grid.dim(10,10);

food = Resource{
    owner: String ("none")
    id: integer}
```

```
agent = Agent_body{
    id: integer(SELF);
    power: integer(50);
    vision: PERCEPTION;
    sense_food: PERCEPTION;
    walk: ACTION;
    attack: ACTION;
    eat: ACTION;
}

vision = PERCEPTION{
    cell[+0][+0].contents; cell[+0][+1].food.owner;
    cell[+0][-1].contents; cell[+1][+0].food.owner;
    cell[-1][+0].contents;
    cell[+0][+0].food.owner; cell[+0][+1].food.owner;
    cell[+0][-1].food.owner; cell[+1][+0].food.owner;
    cell[-1][+0].food.owner;
}

sense_food = PERCEPTION{
    cell[+1][+1].food.id; cell[-1][-1].food.id;
    cell[-1][+1].food.id; cell[+1][-1].food.id;
    cell[+0][-2].food.id; cell[+0][+2].food.id;
    cell[-2][+0].food.id; cell[+2][+0].food.id;
}

eat = ACTION(FOOD_ID:integer){....}
walk = ACTION{....}
attack = ACTION{...}
```

In the code excerpt 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 being capable of two types of perception — vision and food sensing – and being 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 within a 2-cell radius.

For this example, the grid is partitioned in four normative places of equal sizes, and the normative objects are defined and placed in three of the four quadrants, as shown in the code excerpt below:

```
upper-left = PLACE{
type = "food-protected"
environment.grid[0..4][0..4];
}

upper-right = PLACE{
   type = "food-protected"
   environment.grid[5..9][0..4];
}

lower-left = PLACE{
type = "food-protected"
environment.grid[0..4][5..9];
}

norm-obj1 = NORM_OBJ{
    type := "prohibition";
    place:= "upper-left"
    norm := "prohibited(true,attack(SELF,AGENT))";
}

norm-obj2 = NORM_OBJ{
    type := "prohibited";
```

```
    place:= "upper-right"
    norm := "prohibited(not(in-possession(SELF,FOOD)),eat(SELF,FOOD))";
}

norm-obj3 = NORM_OBJ{
    type := "prohibition";
    place:= "lower-left"
    norm := "prohibited(true,attack(SELF,AGENT))";
}

norm-obj4 = NORM_OBJ{
    type := "prohibited";
    place:= "lower-left"
    norm := "prohibited(not(in-possession(SELF,FOOD)),eat(SELF,FOOD))";
}
```

The normative objects in the above example are very simple, and are given simply to illustrate how they can be modelled in our approach. For instance, `norm-obj1` and `norm-obj3` say that an agent ought not to attack (steal food from) another agent, while `norm-obj2` and `norm-obj4` say that the agent ought not to eat a food item which is not in possession of the agent itself.

Clearly, the agents' behaviour will be different in the four quadrants of the environment:

- in the upper-left quadrant, an agent is advised from eating food that is in the possession of another agent, since the situated norm states that an agent is prohibited from stealing food;
- in the upper-right quadrant, agents are supposedly prohibited of doing that, but not effectively, since the situated norm only prohibits the eating of food that is not in the possession of the agent itself (rather than the stealing of food); so, an agent can eat food that previously was in the possession of another agent if it first manages to steal that food;
- in the lower-left quadrant, both restrictions are on; we note that this situation can be seen as redundant, if one understands that the second norm is implied by the first one;
- the remaining quadrant (lower-right) is a lawless area, where agents are completely free to attack each other and to eat anyone else's food.

Notice that `prohibited` is used as a conditional deontic operator, with two arguments: the first argument is a condition to be tested, the second argument is the action that is prohibited. In the modelled problem, the agents are not forced to follow the rules, but the agents use the normative information to monitor each other, using it as input for a reputation system.

6 Related Work

The notion of artifacts [17] and coordination artifacts [13] resembles our notion of *normative objects*. As defined in [13], 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 [17] were presented as a generalisation of coordination artifacts. Artifacts are an abstraction to represent tools, services, objects and entities in 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. The main objective of a coordination artifacts is to be used as an abstraction of an environmental coordination service provided to the agents. However, coordination artifacts express normative rules only implicitly, through their practical effects on the actions of the agents, and so their normative impact does not require any normative reasoning from the part of 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.

Our choice has the advantage of keeping open the possibility of agent autonomy, as suggested in [5]. Agents are, in principle, able to decide whether to follow the norms or not, when trying to be effective in the pursuit of their goals. This is something that is not possible if an agent's action can only happen if in accordance to norms enforced by coordination mechanisms.

Another 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 part of the organisation that is 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 [9], 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 (i.e., groups) and the physical structures (i.e., areas) as "specialisations" of a generic space. The social structures are not contextualised in the space as they are in our work, leaving the social and physical structures quite unrelated.

In ELMS, however, it is not possible to explicitly define social structures, even though it would be possible to implicitly define them through the norms. This is because the aim of ELMS is, as mentioned earlier, to allow for environmental infrastructures compatible with existing approaches to organisational modelling, not for the modelling of organisations as such; the combination of ELMS with existing approaches to modelling organisations is planned as future work.

Another important series of related work is that on Electronic Institutions [10]. The internal working of an electronic institutions is given (in a simplified view) as a state-machine where each state is called a "scene". Each scene specifies the set of roles that agents may perform in it, and a "conversation protocol" that the agents should follow when interacting in the scene. To traverse the series of scenes that constitute the operation of the electronic institution, agents must do a sequence of actions in each scene, and also to commit to certain actions in certain scenes, as the result of their

having performed certain other actions in certain other scenes. Our notion of normative space was inspired by such notion of scene, through giving it a physical, spatial content.

Similar to the electronic institutions approach, there is work on *computational institutions* [15], which are defined as virtual organisations ruled by constitutive norms and regulative norms. In computational institutions, organisational modelling uses the abstraction of coordination artifacts as building blocks, in a way that is very similar to our use of normative objects in spatially distributed organizations, but still keeping implicit in coordination artifacts the normative content imposed on the agents.

7 Conclusions

In this paper we have extended the ELMS 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: some are top-down approaches with focus on the organisations, while bottom-up approaches 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 the notions of *situated norms*, *normative places*, and (spatially distributed) *normative objects*.

It is important to note 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 in which organisations and physical environments are modelled. In future work, with the integration of current means for defining organisational structures with ELMS, and thus with the possibility of associating them to normative places, we hope to contribute to a more integrated approach to designing and implementing the various aspects of multi-agent systems: concentrating on one particular organisation section at a time, specially if it is an organisation section attached to a spatial location, makes it easier for designers to define the groups, roles and agent behaviour that should operate in that particular organisation section.

We believe that an explicit environment description is an important part of a multi-agent system because 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 organisations 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 facilitate the modelling of large organisations.

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 are meant to be followed. It is also more efficient (by taking advantage of natural distribution) to have norms spread in an environment than having them in a repository made available for the whole society, as it is usually the case. Another advantage of having the information distributed is the possibility of the *pre-compilation* of the norms with the spatial con-

text information, what we call *situated norms*. This makes much easiar to make norms operational: to follow and to verify the compliance of the norms. Using the resulting information of the norm verification as input for a reputation system, and having different norms ot each place, even with the same behaviour an agent may have different reputations to each place, what may be a interesting subject to future works.

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. Also as future work, we intend to allow a normative place to be associated with group structures, creating a connection between the organisational structures and the physical environment. We plan to make possible such association for any existing approach to agent organisations, such as $\mathcal{M}$OISE$^+$ [11], OperA/OMNI [16], GAIA [18], or approaches based on electronic institutions [7, 8]. The recursive nature of normative places may not be compatible, however, with some of such approaches to organisation, where the (possibly implicit) system of normative rules has no provision for a recursive structure in its operation.

Acknowledgments

This work was partially supported by CNPq and FAPERGS.

References

1. R. H. Bordini, A. C. d. R. Costa, J. F. Hübner, A. F. Moreira, F. Y. Okuyama, and R. Vieira. MAS-SOC: a social simulation platform based on agent-oriented programming. *Journal of Artificial Societies and Social Simulation*, 8(3), 2005.
2. R. H. Bordini, J. 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 edition, Aug 2005. `http://jason.sourceforge.net/`.
3. R. H. Bordini, J. F. Hübner, and R. Vieira. ***Jason*** and the Golden Fleece of agent-oriented programming. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, 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. C. Castelfranchi, F. Dignum, C. M. Jonker, and J. Treur. Deliberative normative agents: Principles and architecture. In *6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL)*, Lecture Notes In Computer Science, Vol. 1757, pages 364–378, Londo, 1999. Springer-Verlag.
6. R. Conte and C. Castelfranchi. *Cognitive and Social Action*. UCL Press, London, 1995.
7. M. Esteva, D. de la Cruz, and C. Sierra. Islander: an electronic institutions editor. In *AAMAS*, pages 1045–1052. ACM, 2002.
8. M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. Ameli: An agent-based middleware for electronic institutions. In *AAMAS*, pages 236–243. IEEE Computer Society, 2004.

9. J. Ferber, F. Michel, and J.-A. Báez-Barranco. Agre: Integrating environments with organizations. In *E4MAS*, pages 48–56, 2004.
10. A. Garcia-Camino, P. Noriega, and J. A. Rodríguez-Aguilar. Implementing norms in electronic institutions. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *AAMAS*, pages 667–673. ACM, 2005.
11. J. F. Hübner, J. S. Sichman, and O. Boissier. $\mathcal{M}$OISE$^+$: 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.
12. F. Y. Okuyama, R. H. Bordini, and A. C. da Rocha Costa. ELMS: An environment description language for multi-agent simulations. In D. Weyns, H. van Dyke Parunak, and F. Michel, editors, *Proceedings of the First International Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS-04, 19th of July*, number 3374 in Lecture Notes In Artificial Intelligence, pages 91–108, Berlin, 2005. Springer-Verlag.
13. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS'04*, 2004.
14. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. Van de Velde and J. Perram, editors, *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 22–25 January, Eindhoven, The Netherlands*, number 1038 in Lecture Notes in Artificial Intelligence, pages 42–55, London, 1996. Springer-Verlag.
15. R. Rubino, A. Omicini, and E. Denti. Computational institutions for modelling norm-regulated MAS: An approach based on coordination artifacts. In G. Lindemann, S. Ossowski, J. Padget, and J. Vazquez-Salceda, editors, *1st International Workshop "Agents, Norms and Institutions for Regulated Multi-Agent Systems" (ANI@REM 2005)*, AAMAS 2005, Utrecht, The Netherlands, 25 July 2005.
16. J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 11(3):307–360, 2005.
17. M. Viroli, A. Omicini, and A. Ricci. Engineering MAS environment with artifacts. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, pages 62–77, AAMAS 2005, Utrecht, The Netherlands, 26 July 2005.
18. M. Wooldridge, N. R. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.