Norm-Based Behavior Modification in Reflex Agents

Gustavo A. L. de Campos, Emmanuel S. S. Freire, Mariela I. Cortés
Computer Science Department
Universidade Estadual do Ceará (UECE)
Fortaleza, Brazil
gustavo@larces.uece.br, savio.essf@gmail.com, mariela@larces.uece.br

Abstract—**Norms in multi-agent systems are used to regulate the behavior of agents, organizations and sub-organizations in their environments during a period of time. Much of the work on norms focuses on the specification and maintenance of normative system. Little work concentrate on the level of individual agents as the impact of norms on the different types of intelligent agent programs, as the internal modifications in the agents´ decision processes and the background information necessary for rationality in an environment governed by rules. This paper is a contribution in this direction to the case of simple reflex agent based on condition-action rules. An approach to extend this type of agent was formalized and validated it in an implementation of a simple world in the Prolog System. The results demonstrate that the approach is adequate and must be extended to consider other kinds of intelligent agents, as is the case of the goal-based agents and the utility-based agents.**

Keywords- Reflex agent architectures; Deontic concepts; Norms

I. Introduction

Multi-agent Systems (MAS) are becoming an interesting research in computer science area as a paradigm for development and creation of software systems [1] [2]. Long time MAS have been successfully applied to the development of different types of software in academia and industry [1] [2], a fact that encourages the use of this technology in complex systems.

MAS can be understood as societies in which autonomous and heterogeneous entities can work together. The main element of MAS is the agent entity and very different definitions are proposed to its concept. According to [3], an agent is an entity capable of perceiving its environment through sensors and acting upon that environment through actuators. Unlike objects, agents are entities (i) autonomous and not passive, and (ii) able to interact through messaging and not the explicit invocation of a task, as in the case of objects [4].

There are several ways of classifying agents, but the most accepted classifies agents according to their architecture [3]. The agent architecture specifies how is the process of deliberation and choice of action to be taken, according with the agent perceptions [5]. Internal architectures play a central role in the development process of agents, because it is used as a guide that determines its properties, attributes, mental and behavioral components, determining thus a different implementation for each case.

The creation of the agent function, which sets the mapping between the channels of perception and action, is one of the important contributions of Artificial Intelligence. Depending on its function on the environment, the choice of the agent architecture may not be a trivial task. Mainly, the appropriateness of the properties associated with the internal architecture of the agent and the properties of the external environment where the goals should be conducted is crucial.

In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, governance (or law enforcement) systems have been defined. The governance systems define a set of norms (or laws) that must be followed by the entities in the system. Norms provide a means for regulating the agents' behavior by describing their permissions, prohibitions and obligations [6].

Norms are used to regulate the behavior of the agents in MAS by describing the actions that can be performed or states that can be achieved (permissions), actions that must be performed or states that must be achieved (obligations), and actions that cannot be performed or states that cannot be achieved (prohibitions). Thus, norms represent a way for agents to understand their responsibilities and the responsibilities of the others. Norms are used to cope with the autonomy, and are useful to regulate the different interests and desires of the agents that cohabit the system.

As claimed by [7], much of the work developed on norms focuses on the specification and maintenance of normative system, i.e., the research efforts has been concentrated at the macro level of the MAS. Noticing that very little work has been done on the level of individual agents, they proposed some adaption in BDI agents to solve tasks in open environments with the presence of norms of four types.

We believe that there is a demand for similar work for the cases of other different types of intelligent agent programs, that the effort of research must be concentrated on the internal modifications in the agents´ decision processes and the background information, that are necessary for an rational agent in an environment governed by rules. So, this paper is a contribution in this direction to the case of simple reflex agent based on condition-action rules. It describes an adaptation in the internal architectures of these reflex agents so as they can comply with two types of norms.

We formalize the approach employing the Prolog System. It was validated in an implementation of a simplified version of a vacuum cleaner where the agent can be obligated and/or prohibited to perform some actions in the environment. The results demonstrate that the approach is adequate and must be extended to consider other kinds of intelligent agents, as is the case of the goal-based agents and the utility-based agents.

The paper is structured as follows. Section 2 briefly presents the reflex agent architecture. The concepts related to norms are detailed in Section 3. Section 4 describes how to combine rightly the use of reflex agent architectures with norms. A case study is showed in Section 5 and, finally, conclusions and future works are discussed in Section 6.

II. Simple Reflex Agent

The internal architectures of agents can be categorized based on reactive and proactive foundations. A reactive agent must attend continuously to changes in their environment, through the selection of actions, typically based on the current perception of the environment and additional knowledge about the possible actions that are best suited to different conditions in which the environment can be.

A proactive agent can take the initiative and select actions from a set of possible atomic actions or plans, and create sequences of actions that attain goals in the environment task. In this context, [3] describe the principles underlying almost all intelligent systems through programs for reactive and proactive agents. This paper focuses norms related to reflex agent, thus, the main characteristics of the simple reflex agents are briefly presented below.

According to [3], a simple reflex (or reactive) agent is the simplest type of agent. This architecture assumes that at any time, the agent perceives information about the state of the environment through sensors and based on rules in the form “if condition then action”, it selects the most adequate action for the current perception. The agent performs the selected action upon the environment through actuators. The Figure 1 presents a schematic diagram of the simple reflex agent, synthesizing the Russell&Norvig´s ideas related to reactive agent program, as well as the abstract architecture of the this agent, which was proposed by Wooldridge in [8].

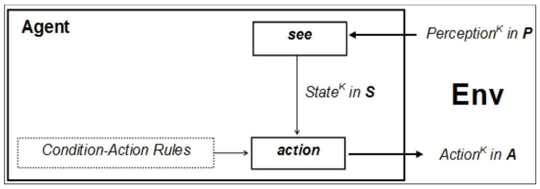

Figure 1. Schematic diagram of the simple reflex agent

This synthesis assumes that at any instant K:

(1) through sensors, the agent receives information from the environment, i.e., perceptions defined on a set, P = {Perception1, ..., Perceptionn}, of n possible perceptions from the environment (Env);

(2) a perception subsystem, see: P $\rightarrow$ S, that processes each perception in P and maps to one of m possible states, S = {State1, ..., Staten}, that are representations of aspects in the perceptual information that are accessible to the agent;

(3) a subsystem for decision making, action: S $\rightarrow$ A, that processes the states in S and selects, according to an specific rule of the set of condition-action rules, one of the l actions in the set of possible actions for the agent, A = {Action1, ..., Actionl};

(4) through actuators, the agent sends the selected action for the environment;

(5) in the interaction K+1, the agent initiates another cycle involving the perception of the world through the function see and the selection of an action to be executed by the function action.

The condition-action rules consist in supplemental information that the agent uses during the decision making process. They can be seen as a set of common associations which are observed between certain conditions established from the descriptions of States in S and certain Actions in A. The information about the agent´s goals is not explicitly considered in this architecture, but implicitly in the condition-action-rules. Thus, the agent´s designer defines these rules having in mind the performance measure that will be applied to the agent. In this context, it is expected that, in an adequate environment, if the rules are adequate, then the agent will achieve its objectives and, consequently, it will be well evaluated.

III. Norms for Agents

The norms are used to restrict and guide the behavior of agents, organizations and sub-organizations during a period of time. In this sense, a norm includes a set of sanctions applied to the entities that violated or fulfilled the norm [9]. In this section, the main elements that compose the norm are explained considering the types of norms and their representation.

A. Elements of Norms

Bellow we describe the main elements which compose a norm, based on a survey of existing specification and implementation languages for norms [9].

- Deontic Concepts: the deontic logic refers to the logic of requests, commands, rules, laws, moral principles and judgments [10]. In multi-agent systems, such concepts have been used to describe the constraints for the behavior of agents in the form of obligations (what the agent must execute), permissions (what the agent can execute) and prohibitions (what the agent cannot execute).
- Involved Entities: considering that the norms are defined to restrict the entities´ behavior, the identification of related entities is essential. The norm may regulate the behavior of individuals (for example, a particular agent, or an agent, while playing a particular role), or the behavior of a group of individuals (for example, all agents playing a particular role, groups of agents, groups of agents playing roles or all agents in the system).
- Actions: once a norm is set to restrict the behavior of the entities, it is important the clear specification of the actions that are being regulated. Such actions may be communication, usually represented by sending and

receiving a message, or non-communicative actions (such as access and modify a resource, get in an organization, move to another environment, etc.).

- Activation Constraints: a norm have a period of time in which its restrictions must be fulfilled, but only when this norm, is active. Norms may be activated by a constraint or a set of constraints that can be: the execution of actions, the definition of specific time intervals (before, after or in between), the reaching of system states or temporal aspects (such as dates) and also the activation/deactivation of other norm and fulfillment/violation of a norm.
- Sanctions: when a norm is violated the entity may suffer a punishment, and when a norm is fulfilled, the entity involved may receive a reward. Rewards and punishments are referred to as sanctions and should be related to the norm specification.
- Context: the norms are usually defined in a determined context that determines the application area. The norm may, for example, be described in the context of a specific environment and must be filled only by agents in execution in the environment. Similarly, a norm can be defined in the context of an organization and fulfilled only by the agents that play a role in the organization.

B. *Norm Types and Representation*

This paper considers two types of norms of four possibilities discussed in [7]. The classification scheme considers whether norms are obligations or prohibitions, and whether they refer to states of the world or to particular actions. In this context, Table 1 presents the four types of norms obtained and a description about what the agent should do when accepting a norm type.

TABLE I. THE TYPES OF NORM AND THEIR DESCRIPTIONS

Norms Types		**Description**
1	obligation(***p***)	agent must try to achieve certain world state ***p***
2	obligation(***a***)	agent must try to execute certain actions ***a***
3	prohibition(***p***)	agent must try to refrain from achieving a state ***p***
4	prohibition(***a***)	agent must try to refrain from executing an action ***a***

Each type of norm has activation and expiration conditions as, for instance, a well defined validity period of time, indicating when the norm is in force and when it ceases to be in force. In our approach, as [7], we leverage representational concepts from the formalization of [11]. This formalization includes notions of activation and expiration of a norm: norm(Activation, Expiration, Norm), where Activation is the activation condition for the norm to become active, Expiration is the expiration condition to deactivate the norm, and Norm is the norm itself.

It is important to note that we are not concerned, at this point, with handling more complex norm representation schemes. Moreover, in order to facilitate the creation of concrete agent behaviors, in compliance with a set of norms, this paper approaches only norms of Types 2 and 4.

IV. MODIFYING REFLEX AGENTS TO ADOPT NORMS

In this section we examine how norms can influence the choice of actions according to the internal architecture of a simple reflex agent.

A. *An Outline of the Approach*

Purely reflex agents should be able to quickly respond to changes in the environment. This kind of agent may be inserted into an environment that has a specified set of norms that restrict their actions. As defined by [9], the norms are intended to restrict the behavior of agents applying sanctions when they are violated or fulfilled. Therefore, the norms of an environment should not be able to avoid the execution of certain action, but rather to penalize or reward an agent if the action taken by it is prohibited or obligated. Therefore, if the set of norms defined in the environment is not considered in the condition-action rules, an agent can be penalized if it performs a prohibited action.

In order to avoid the violation of the simple reflex agent architecture, our approach proposes to consider the information about the set of norms as an extension of the condition-action rules. It involves the definition of three different groups of condition-action rules. Each group is associated with one deontic concept and considers the sanctions linked to each norm, that is:

- Obligation Rules Group: specifies the rules related with the actions that must be performed by the agents. If an event of environment matches with a rule in this group, it must necessarily be performed by the agent;
- Prohibition Rules Group: specifies the rules that are related with the actions that cannot be performed by the agent. If an event of environment matches with a rule in this group, the rule will not be executed by the agent;
- Permission Rules Group: specifies the rules related with the actions that can be executed. If an event of environment matches a rule set out of this group, it may or may not be executed by the agent.

Figure 2 shows the schematic diagram of the simple reflex agent in an environment with norms. We added two new groups in the agent´s action selection mechanism, corresponding to the representation of the information about its obligations and prohibitions. Our approach considers that if an action is obligated, then the agent must perform that action only if it is not prohibited. If an action is prohibited, then the agent must perform another action, different from the prohibited action, which is permitted and rational. If there is not an action that is obligated and prohibited, then the agent must perform a permitted action which is rational, as would do a well designed simple reflex agent in an environment without norms.

The function Simplex-Reflex Agent outlined in the sequence of five steps in Section 2 must be adapted to become valid in the case norms were activated. In order to avoid conflicts between rules from the different groups, Step 3 of the sequence must respect the following rules:

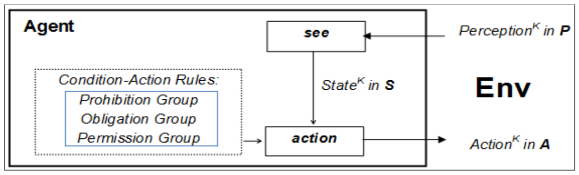

Figure 2. Schematic diagram of the simple reflex agent with norms

(3.1) first, it searches for rules in the Obligated Rules Group to find those actions that must be performed in the environment and are not prohibited;

(3.2) if there is some prohibited action, then the function inhibits rules in the Obligated Rules Group and searches for rules in the Prohibited Rules Group to find those actions that are not prohibited and can be performed, according with the environment estate conditions;

(3.3) if there are not any prohibition, the function selects the action that must be performed as indicated by the obligation norm; and

(3.4) finally, in the case where there not exist any obligations and prohibitions, the function searches for rules in the Permission Rules Group in order to find some action that can be performed, according with the environment estate conditions.

The strategy to select the action that is implicit in the approach considers that the rational behavior is achieved when the agent is able to maximize the rewards that are consequences of: (a) the selection of an obligatory and not prohibited action; (b) the non-selection of a prohibited action, and (c) the selection of permitted actions, those which are adequate with the environment state conditions and with the agent´s performance measure.

B. Norms Outcomes

In this section, we consider the agent accepts a norm immediately as it perceives it and that the behavior modification must ensue according to activation and expiration conditions of the norm. More specifically, the required change in the behavior of the agent that accepts a particular norm should occur according with the conditions of activation and expiration. For instance, in the case of an agent that is goal oriented, it can be necessary that the agent add the goal of achieving a state p, when a particular norm of obligation is activated, and then, when the norm is expired, delete this goal.

In the case of simple reflex agents, Tables 2 and 3 summarize all norm combinations of activation conditions (AC) and expiration condition (EC), and their outcomes respectively for norms Types 2 and 4, considered in this paper and highlighted in Table 1. The AC and EC columns give the truth-value of these conditions at the time the agent accepts the norm.

When both AC and EC are true (row 1 in Tables 2 and 3) results in the norm being ignored, as its expiration condition has already elapsed. When AC is false but EC is true (row 4 in Tables 2 and 3), norms are also ignored since, again, they have already expired. When an agent accepts an obligation, if the AC is already true (row 2 in Table 2) it must react to execute the action specified in an obligation(a) if the action is not specified in an prohibition(a). While for a prohibition (row 2 in Table 3), it must refrain from executing the offending action, and try to execute an action that is permitted and adequate to the environment conditions and to its performance measure.

TABLE II. NORM TYPE 2 - OBLIGATION(A).

AC	EC	Outcome
True	True	**(1)** Ignore norm
True	False	**(2)** Execute action ***a*** if ***a*** is not prohibited; otherwise execute an action that is not prohibited, but is permitted and adequate
False	False	**(3)** Ignore norm
False	True	**(4)** Ignore norm

TABLE III. NORM TYPE 4 – PROHIBITION(A).

AC	EC	Outcome
True	True	**(1)** Ignore norm
True	False	**(2)** Suppress rules in the Permitted Group that include action ***a*** and execute an adequate permitted action
False	False	**(3)** Ignore norm
False	True	**(4)** Ignore norm

We are considering, the agent must ignore the norm, if AC and EC are false (row 3 in Tables 2 and 3). We are considering that in the situation that the norm has not yet been activated, the agent has already been programmed for the activation of the norm in the future. Therefore, we are not considering the situation that the agent is able to include the proposed scheme at the moment it perceives a norm, that is, when both AC and EC are false.

C. Formal Description of the Approach

This section presents, first, a declarative description of the reflex agent program shown in Figure 1, that is, in an environment without norms and, second, an extension of this description for the case in which the agent perceives norms Types 2 and 4, as shown in Figure 2. The Prolog language was employed to formally describe the agent programs in both cases. The Prolog definition agent/2 describes the agent program discussed in Section 2:

```
agent(P,A):- see(P,S), action(A).
```

where the definition see/2 implements the perception subsystem, as indicated by Step 2 of the sequence of five steps illustrated in Section 2, it inserts, in the Prolog´s base, some predicates employed to represent the environment state; and the definition action/2 implements the decision-making subsystem of the agent program, as indicated by Step 3 in the sequence, i.e.:

```
action(A):- do_permission(A),!.
```

The definition do_permission/1 implements the Permission Rules Group for the agent in an environment without rules, which are condition-action rules highlighted in Figure 1. As mentioned, these rules depend of the agent´s task environment and must be defined according with the available knowledge about the actions that are most adequate to be executed when the environment is under certain conditions. The next section of this paper illustrates this group of rules for the case where the agent is a vacuum cleaner in a very simple environment.

As the Subsection 4.1 indicates in the extension of Step 3 in Section 2, that is, Steps 3.1-3.4, in the case there are norms in the environment, the definition do_permission/1 will be accessed by the decision making subsystem only when active norms of Type 2 are not available. When some active norm is available, the action must be executed immediately if it is not a prohibited action by a norm of Type 4. If some action is a prohibited action, the rules in the do_permission definition must be accessed in order to indicate for the agent those not prohibited actions that are adequate with the environment conditions.

So, in order to implement the above procedure, our approach proposes that, instead of access exclusively the rules in do_permission/1, the decision making subsystem considers the three groups of rules illustrated in Figure 2 and the sequence of steps 3.1-3.4. The definition do/1 below consists of an extension in the action function, i.e:

```
action(A):- do(A),!.
```

to consider these new possibilities:

```
do(A):- do_obligation(A).
do(A):- do_prohibition(A).
do(A):- do_pemission(A).
```

where the definitions do_obligation/1 and do_prohibition/1 can be declared without concern with a specific domain and are related, as we saw that the agent will execute an action which is obligated only if it is not prohibited.

The three rules bellow compose the Obligation Group and are in accordance with what was discussed in the Section 4.2 and, more specifically, in the Table 2, that identifies the changes that are necessary to a simple reflex agent program for the case in which the environment imposes norms of Type 2 for the agents:

```
do_obligation(A):-
            norm(AC1,EC1,obligation(Ac)),
            is_True(AC1), is_False(EC1),
            norm(AC2,EC2,prohibition(Ac)),
            is_True(AC2), is_False(EC2),!,
            do_prohibition(A).
do_obligation(A):- norm(AC,EC,obligation(A)),
                   is_True(AC), is_False(EC),!.
do_obligation(A):- norm(_,_,obligation(_)),
                   do_prohibition(A).
```

where the predicates is_True/1 and is_False/1 must be specified according, respectively, with the activation or expiration conditions being considered in the arguments of the predicates stating norms.

The two first rules of the above definition are related to the situation described in row 2 of Table 2, that is, in which the activation condition (AC1) is True and the expiration condition (EC1) is False in some norm of Type 2. The first rule considers the case in which another norm of Type 4 is in the same situation (AC2 = True and EC2 = False) and the action which is obligated is, simultaneously, prohibited. The outcome of this situation is that the agent must perform an action (A) that is not prohibited, but is permitted and adequate.

The second rule considers the case in which none norm of Type 4 is active, that is, in which the agent must perform the action that is obligated by the active norm of Type 2. Finally, the third rule in the definition is related with the rows 1, 3 and 4 of Table 2. In these situations the agent must ignore the obligation norm and perform an action that is permitted and adequate, according with the conditions of the environment and the rules in the definition do_permission/1.

The two rules bellow compose the Prohibition Group and are in accordance with what was discussed in the Section 4.2 and, more specifically, in the Table 3, that identifies the changes that are necessary to a simple reflex agent program for the case in which the environment imposes norms of Type 4 for the agents:

```
do_prohibition(A):- norm(AC,EC,prohibition(Ac)),
                    is_True(AC), is_False(EC),!,
                    do_permission(A), not(A=Ac),!.
do_prohibition(A):- norm(_,_,prohibition(_)),
                    do_permission(A),!.
```

The first rule of the above definition are related to the situation described in row 2 of Table 3, that is, in which the activation condition (AC) is True and the expiration condition (EC) is False. The outcome of this situation is that the agent must perform an action (A) different from the prohibited action (Ac), but which is permitted and adequate, according with the conditions of the environment and the rules in the definition do_permission/1.

The second rule in the definition is related with the rows 1, 3 and 4 of Table 3. In these situations the agent must ignore the prohibited norm and performs an action that is permitted and adequate. So, this last rule completes the second group of rules which the approach supposes be a necessary modification to produce a rational behavior of a simple reflex agent in an environment with the presence of norms of the Types 2 and 4.

V. Case Study

This section describes the third group of rules, Permission Rules Group, for a very simple problem, but still very useful to illustrate the ideas discussed in the last section. We specify the rules in the definition do_permission/1 for the case in which the agent is a vacuum cleaner in a world containing only two rooms. So, employing this specific definition, we used the Prolog System to: (1) describe experiments involving the agent in the world without and with the presence of norms, (2) record the history of the vacuum cleaner in the world, and (3) measure its performance.

Considering the vacuum cleaner world with only two rooms, where each room can be clean or dirty, in our experiments the perceived information of the environment were represented by the following atoms in Prolog: roomA, roomB, clean and dirty. So, we assume the existence of eight possible perceptions for the environment (P), which were represented by eight lists of three atoms in Prolog, i.e.:

```
P = {[roomA,dirty,dirty], [roomA,dirty,clean],
     [roomA,clean,dirty], [roomA,clean,clean],
     [roomB,dirty,dirty], [roomB,dirty,clean],
     [roomB,clean,dirty], [roomB,clean,clean]}.
```

Additionally, we assume that the agent perception is local, i.e., see/2 perceives only the room where it is located and the state of the room. So, we assume the existence of four possible internal states for the environment (S), which were represented by four lists of two atoms in Prolog, i.e:

```
S = {[roomA,dirty], [roomA,clean],
     [roomB,dirty], [roomB,clean]}.
```

In each of the eight states of the world (P), there are three possible actions for the vacuum cleaner (A), which were represented by three atoms in Prolog, i.e:

```
A = {suck, right, left}.
```

In the beginning of each experiment the vacuum cleaner does not know the world configuration in terms of dirt. We considered that when the world is without the presence of norms, the measure of performance evaluation offers the reward of one point per each square clean (+1) and penalizes with the loss of one point per each movement (-1). In the case of the presence of norms in the world, the measure must be adapted in order to consider the rewards (+points) and the penalties (-points), which are consequences of the agent accepting or rejecting some norm.

A. Environment without Norms

In the Case 1, there are no norms in the world. The Prolog definition below describes the simple reflex vacuum cleaner:

```
vacuum_cleaner(P,A):- see(P,S), action(A).
```

where the terms P, S and A were defined generically in Section 2 and in the specification of the environment properties outlined in the introduction of this section; see/2 has been specialized to deal with the atoms and lists employed to represent the sets P and S; action/1 has been specialized to deal with the set A and with the specific rules do_permission/1 for this world, i.e:

```
action(A):- do_permission(A),!.
```

The definition do_permission/1 implements the Permission Rules for the agent in an environment without norms, are the condition-action rules highlighted in Figure 1. For this Case 1, was sufficient to generate a definition with four Prolog phrases, i.e:

```
do_permission(suck):- in(Romm),  is(Room,dirty).
do_permission(right):- in(roomA).
do_permission(left):-  in(roomB).
do_permission(no_op).
```

where the predicates in/1 and is/2 were used to represent the conditions in the antecedents of the rules and are known by the agent at the moment when the agent perceives the environment by see/2. Table 4 highlights the episodes performed by vacuum_cleaner/1 in six interactions with an environment without norms where, initially, the agent is in room-A, and room-A and-B are dirty.

The two first columns in the table identify the perceptions and actions of the agent in each interaction. The third and fourth columns identify the values of performance measure per episode (Ep) and history (H). Since the perception is local and at the agent does not have an internal state to avoid unnecessary movements (interactions four to six), there were no surprises in the behavior of the agent. The set of rules do_permission/1 provided a rational behavior for the agent, except for the unnecessary movements that caused a negative performance evaluation.

TABLE IV. PERFORMANCE IN A WORLD WITHOUT NORMS

P	A	Ep	H
[roomA,dirty,dirty]	suck	1	1
[roomA,clean,dirty]	right	-1	0
[roomB,clean,dirty]	suck	1	1
[roomB,clean,clean]	left	-1	0
[roomA,clean,clean]	right	-1	-1
[roomB,clean,clean]	left	-1	-2

B. Environment with Norms

In the second case (Case 2), the environment is governed by norms of the Types 2 and 4. Consider that the measure of evaluation should apply the appropriate sanctions, as the fulfillment or not of the established norms. In this particular example, the measure offers with reward three-point (+3) for the fulfillment of an obligation and two points (+2) for the fulfillment of a prohibition. In any other situation, the measure behaves in accordance with the specification realized for the Case 1. In our experiments, first, we considered a norm of Type 2 in which the environment requires the execution of a specific action by the vacuum cleaner during a period of time. For instance, the predicate below states that "The agent must suck the room-A from 4:00 to 6:00 a.m.":

```
norm(roomA, time([4,6]), obligation(suck)).
```

where the first argument of the statement identifies the condition of activation (AC), the second identifies the condition of expiration (EC), and the third is the action to be performed.

The definition do_obligation/1, which implements the Obligation Rules, are necessary for the vacuum cleaner in a simplified environment regulated by the norm stated as above. Assuming that no prohibition norm is present, the first rule in the definition do_obligation/1 will never be activated. However, the other two rules will be activated according to the truth value of AC and EC, which are obtained by the evaluation of the predicates is_True/1 and is_False/1 specifically for the above obligation norm in the vacuum cleaner world:

```
is_True(Room):-  in(Room).
is_False(time[T1,T2]):- clock(Hour),
                        Hour >= T1, Hour =< T2.
```

where in/1 is the same predicate which the agent employs to represent where it is located and clock/1 is a predicate which represents the agent´s simplified clock in the experiments.

Table 5 highlights the episodes performed by the vacuum cleaner in seven interactions with an environment with a norm of Type 2, as stated above.

TABLE V. PERFORMANCE IN A WORLD WITH NORM OF TYPE 2

P	A	Ep	H
[roomA,dirty,dirty]	suck	1	1
[roomA,clean,dirty]	right	-1	0
[roomB,clean,dirty]	suck	1	1
[roomB,clean,clean]	left	-1	0
[roomA,clean,clean]	suck	3	3
[roomA,clean,clean]	suck	3	6
[roomA,clean,clean]	right	-1	5

The rows one to four describe the behavior of the agent in a period in which the norm had not been activated and the agent was governed by the rules in the Permission Group. The rows five to six of the table (shaded) illustrate the behavior of the agent when the norm of the type 2 was activated (AC = True and EC=False). It is noticed that the agent was rewarded with three points per action during the period, according with the sanctions associated with the norm of obligation. In row 7, the norm was expired and the agent behavior was again governed by the Permission rules. These ideas can be extended to the case where the environment is also governed by norms of prohibition. For instance, consider the predicate below states that "The agent cannot suck the room-A from 1:00 to 5:00 a.m.":

norm(room(roomA), time([1,5]), prohibition(suck)).

Table 6 highlights the episodes performed by the vacuum cleaner in seven interactions in the environment with both types of norms.

TABLE VI. PERFORMANCE IN A WORLD WITH NORMS OF TWO TYPES

P	A	Ep	H
[roomA,dirty,dirty]	right	2	2
[roomB,dirty,dirty]	suck	1	3
[roomB,dirty,clean]	left	-1	2
[roomA,dirty,clean]	right	2	4
[roomB,dirty,clean]	left	-1	3
[roomA,dirty,clean]	suck	3	6
[roomA,clean,clean]	right	-1	5

The rows one and four of the table describe the behavior of the agent in a period in which the norm of Type 4 was activated (AC = True and EC=False). In row four, simultaneously with the norm of prohibition, the norm of Type 2 was activated too. It is noticed that in both case the agent was rewarded with two points, according with the sanctions associated with the norm of prohibition, which has priority on any norm of obligation. In row six only the norm of Type 2 was activated and the agent was rewarded with three points, according with the sanctions associated with the norm of obligation. The four remaining rows illustrate the cases where the two norms had expired and the agent´s behavior was governed by the rules in the Permission Group. All rows in the table that refer to the situation in which the two norms were not activated, the vacuum cleaner was evaluated according to the performance measure adopted for the agent in the world without norms.

VI. CONCLUSIONS AND FUTURE WORKS

In this work we discuss the influence of the norm concepts related to the reflex agent architectures in order to improve the performance of the agents executing in an environment governed by norms. In the original conception of reflex architecture, only permission rules are considered. Now, in the proposed approach two new sets of condition-action rules are incorporated in order to define prohibition and obligation rules. Additionally, the logic to implement the behavior of the simple reflex agents is presented. The strategy selects the actions aiming to minimize the penalties and to maximize the rewards. The case study simulates the behavior of a simple reflex agent in a vacuum cleaner word governed by obligation and prohibition rules. In this simple scenario, the proposed approach involving the implicit definition of norms as condition-action rules provides a rational behavior in consistency with the agent architecture specifications. Future works include the analysis of the agent behavior for another agent architectures in the literature, considering the influence of norms in the decision making process so that the agents can understand their responsibilities and the responsibilities of the others.

REFERENCES

[1] Lind, J., 2001. Issues in agent-oriented software engineering, In: P. Ciancarini e M. Wooldride (Eds.) Agent-Oriented Software Engineering, LNCS 1957, Germany, Springer, p.45-58.

[2] Wooldridge, M. and Ciancarini, P., 2001. Agent-Oriented Software Engineering: the State of the Art, In: M. Wooldridge and P. Ciancarini (Eds.), Agent-Oriented Software Engineering, LNCS 1957, Berlin: Springer, p. 1-28.

[3] Russell, S. and Norvig, P., 2003. Artificial Intelligence: A Modern Approach, 2nd Ed., Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2.

[4] Wagner, G., 2003. The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Information Systems, v. 28, n.5, pp. 475–504.

[5] Wooldridge, M. and Jennings, N. R., 1995. Intelligent Agents: Theory and Practice. Knowledge Engineering Review, Vol. 10, No. 2. Cambridge: Cambridge University Press, 1995.

[6] López y López, F., 2003. Social Power and Norms: Impact on agent behavior. PhD thesis, Univ. of Southampton, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science.

[7] Meneguzzi, F. and Luck, M., 2009. Norm-based behavior modification in BDI agents. In: 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary.

[8] Weiss, G., 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press.

[9] Silva, V. T. , Braga. C. and Figueiredo, K., 2010. A modeling language to model norms. The International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2010), 9th. Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canadá.

[10] Meyer, J. J. and Wieringa, R. J., 1993. Deontic logic in computer science: normative system specification, Deontic logic in computer science: normative system specification, John Wiley and Sons Ltd. Chichester, UK.

[11] Oren, N., Panagiotidi, S., Vazquez-Salceda, J., Modgil, S., Luck, M., and Miles, S., 2008. Towards a formalisation of electronic contracting environments. In Proc. 12th COIN Workshop, pages 61–68.