Implementing Norms That Govern Non-dialogical Actions

Viviane Torres da Silva*

Departamento de Sistemas Informáticos y Computación – UCM, Spain, Madrid
viviane@fdi.ucm.es

Abstract. The governance of open multi-agent systems is particular important since those systems are composed of heterogeneous, autonomous and independently designed agents. Such governance is usually provided by the establishment of norms that regulate the actions of agents. Although there are several approaches that formally describe norms, there are still few of them that propose their implementation. In this paper we propose the implementation of norms that govern non-dialogical actions by extending one of the approaches that regulate dialogical ones. Non-dialogical actions are not related to the interactions between agents but to tasks executed by agents that characterize, for instance, the access to resources, their commitment to play roles or their movement into environments and organizations.

Keywords: Norm, governance of multi-agent system, non-dialogical action, implementation of norm.

1 Introduction

The governance of open multi-agent systems (MAS) copes with the heterogeneity, autonomy and diversity of interests among agents that can work towards similar or different ends [9] by establishing norms. The set of system norms defines actions that agents are prohibited, permitted or obligated to do [1] and [12].

Several works have been proposed in order to define the theoretical aspects of norms [3] and [5], to formally define those norms [2] and [4], and to implement them [7], [8], [9], [10] and [13]. In this paper we focus on the implementation of norms. Our goal is to present an approach where dialogical and non-dialogical norms can be described and regulated. Non-dialogical actions are not related to the interactions between agents but to tasks executed by agents that characterize, for instance, the access to resources, their commitment to play roles or their movement in environments and organizations. From the set of analyzed proposals for implementing norms, few approaches considers non-dialogical actions [9], [10] and [13]. Although, the authors present some issues on the verification and enforcement of norms, they do no demonstrate how such issues should be implemented. Other approaches such as [7] and [8] deal with e-Institutions and, thus, consider illocutions as the only action performed in such systems.

* Research supported by the Juan de la Cierva programa, Comunidad de Madrid S-0505/ TIC-407 and MEC-SP TIC2003-01000.

J.S. Sichman et al. (Eds.): COIN 2007 Workshops, LNAI 4870, pp. 232–244, 2008.
© Springer-Verlag Berlin Heidelberg 2008

Our approach extends the work presented in [8] with the notion of non-dialogical actions proposed in [13]. A normative language is presented in [8] to describe illocutions (dialogical actions) that might be dependent on temporal constraints or the occurrence of events. We have extended the normative language in order to be possible to specify non-dialogical norms that state obligations, permissions or prohibitions over the execution of actions of agents' plans (as proposed in [13]) and of object methods. Similar to the approach presented in [8], we have also used Jess[1] to implement the governance mechanism that regulates the behavior of agents. The mechanism activates norms and fires violations (Jess rules) according to the executed (dialogical or non-dialogical) actions (Jess facts).

Although both the normative language and the implementation rules can be used by agents and by the governance mechanism, the approach focuses on the implementation of norms from the system perspective [13], i.e., both agents and the governance mechanism will use the language and the rules to find out: What are the activated and deactivated norms? What are the fulfilled and the violated norms? What are the applied sanctions?

The paper is organized as follows. Section 2 describes the example we are using to illustrate our approach. Section 3 intends to clearly present the difference between dialogical and non-dialogical actions. Section 4 points out the main concepts of the extended normative language and Section 5 describes the implementation of the governance engine in Jess. Section 6 concludes our work and presents some future work.

2 Applied Example

In order to exemplify our approach, we have defined a set of six norms that govern a simplified version of a soccer game. The soccer game is composed of agents playing one of the three available roles: referee, coach and player (kicker or goalkeeper). The responsibilities of a referee in a soccer game are: to start the game, stop it, check the players' equipments and punish the players. The available punishments are: to show a yellow card, send off a player, and declare a penalty. The possible actions of a player during a game are: kick the ball and handle the ball. The coach role is limited to substitute players. Besides those actions, all agents are able to move and, therefore, enter and leave the game field. The six norms that regulate our simple soccer game are the following:

Norm 1: *The referee must check the players' equipments before starting the game.*
Norm 2: *A coach cannot substitute more than three players in the same game.*
Norm 3: *Players cannot leave the game field during the game.*
Norm 4: *The referee must send off a player after (s)he has done a second caution in the same match.* In this simplified version of the soccer game, there is only one situation that characterizes a caution; a player leaving the game field before the referee has stopped it. At the first caution, the agent receives a yellow card.
Norm 5: *Kickers cannot handle the ball.*
Norm 6: *The referee must declare a penalty if kicker handles the ball.*

[1] Jess is a rule-based system. http://www.jessrules.com/

3 Dialogical and Non-dialogical Actions

Non-dialogical actions are the ones not related to interactions between agents. Not all actions executed by agents in MAS provide support for sending and receiving messages between them [13]. There are actions that modify the environment (for example, updating the state of a resource) that do not characterize a message being sent to or received from another agent. In the soccer game example, the actions of kicking the ball or handling it are non-dialogical actions. In addition, actions that modify the position of an agent in an environment do not characterize a dialogical action either. The actions of entering or leaving the game field are not dialogical ones.

Some actions can be defined as a dialogical or a non-dialogical one, depending on how the problem is modeled. In the soccer game, to start a game and to stop it was considered dialogical actions. Agents receive a message informing about the state of the game. The dialogical actions of the soccer game example are: to start the game, stop it, punish player, declare penalty and show the yellow card. The non-dialogical ones are: enter in the game field, leave it, handle the ball, kick the ball, substitute a player and check the player's equipment.

4 Describing Norms

Since our intention is to contribute to the work presented in [7], we extend the BNF normative language to represent non-dialogical actions and to describe conditions and time situations that are defined by those non-dialogical actions. In addition, the specification of dialogical actions already presented in the previous normative language was extended in order to be possible to describe messages attributes stated in the FIPA ACL language[2].

4.1 Specifying Non-dialogical Actions

The original BNF description of the normative language defines norms as the composition of a *deontic* concept (characterizing obligation, prohibition or permission) and an action followed by a temporal situation and an *if* condition, when pertinent. In such definition, actions are limited to utterance of illocutions.

In our proposed extension, the *action* concept was generalized to also describe non-dialogical ones. Dialogical and non-dialogical actions are complementary, as illustrated by the grammar that specifies that these are the only two possible actions' kinds. Non-dialogical actions state the entities whose behavior is being restricted and the actions that are being regulated. Due to the way the *entity* concept was defined, a non-dialogical norm, i.e., a norm that regulates non-dialogical actions, can be applied to all agents in the system, to a group of agents, to agents playing a given role or even to a unique agent.

```
<norm> ::= <deontic_concept> '(' <action> ')'
  | <deontic_concept> '(' <action><temporal_situation> ')'
  | <deontic_concept> '(' <action> IF <if_condition> ')'
  | <deontic_concept> '(' <action> <temporal_situation> IF <if_condition> ')'
<deontic_concept> ::= OBLIGED | FORBIDDEN | PEMITTED
```

[2] http://www.fipa.org/repository/aclspecs.html

```
<action>::= <non_dialogical_action> | <dialogical_action>
<non_dialogical_action> ::= <entity> 'EXECUTE' <exec>
<entity>::= <agent>':'<role> | <role> | <agent> | <group> | 'ALL'
```

In this paper we are limiting non-dialogical actions to the execution of an object/class method or to the execution of the action of an agent plan [13]. Non-dialogical norms that regulate the access to resources specify the entities that have restricted access to execute the methods of the resource. Non-dialogical norms that regulate (non-dialogical) actions not related to the access to resources describe entities that have restricted access to the execution of an action of a plan.

```
<exec> ::= <objectORclass>'.'<method>'('<parameters>')''('<contract>')'
 | <plan>':'<action>'('<parameters>')''('<contract>')'
 |...!the parameters and the contract can be omitted
```

In [13], the authors affirm that non-dialogical actions can be described as abstract actions that are not in the set of actions defined by the agents or in the set of methods of the classes. Agents must translate the actions and methods to be executed into more abstract ones. With the aim to help agents in such transformation, we propose the use of contracts. A contract is used to formally describe the behavior of the actions/methods while specifying its invariants, pre and post-conditions [11]. We do not impose any language to be used to describe the terms of a contracts[3].

```
<contract> ::= <pre>';'<post>';'<inv> |... !pre, post and inv can be omitted
<pre> ::= <expression> | <expression> <opl> <pre> ...
<opl> ::='AND' | 'OR' | 'XOR' | 'NOR'|... !pre, post and inv are similarly defined
```

Such extensions make possible to describe, for instance, norms that regulate the execution of an action while describing the parameters required for its execution and the contract that defines it. The extensions enable, for example, the definition of *norm 2*. Such norm states that a coach cannot substitute more than three players in the same game. The coach cannot execute an action that substitutes players if the number of substitutions is already 3.

```
FORBIDDEN ( coach EXECUTE managingTeam:SubstitutePlayer (outPlayer,inPlayer,team)
           ( team.coach = coach; team.substitutions = team.substitutions@pre+1 AND
             team.playersInField->excludes(outPlayer)AND
             team.playersInField->includes(inPlayer); )
   IF team.substitutions >= 3 )
```

The action governed by *norm 1* is also a non-dialogical action and states that the referee must check the players' equipment before starting the game. The action of checking the equipment is a non-dialogical action since the referee needs not to interact with the player but with its equipment. On the other hand, the action of starting a game is a dialogical action modeled as a message from the referee to everybody in the game (as will be presented in Section 4.4).

```
OBLIGED ( referee EXECUTE managingGame:checkEquipment (players)
   BEFORE ( UTTER(game; s_i; INFORM(;referee;;[;gameStart;;;;;;])))  )
```

[3] In this paper we are using OCL (http://www.omg.org/technology/documents/formal/ocl.htm)

4.2 Extending the Temporal Situations

The *temporal situation* concept specified in the normative language is used to describe the period of valid (or active) norms. Norms can be activated or deactivated due to the execution of an (dialogical or non-dialogical) action, to the change in the state of an object or an agent, to the occurrence of a deadline, and to the combination of such possibilities. In the previous normative languages the authors only consider the execution of dialogical actions and the occurrence of a deadline as temporal situations. The normative language was extended to contemplate the activation and deactivation of norms due to the execution of non-dialogical actions, to the change in the state of an object or an agent (without specifying the action that was responsible for that) and to the combination of the above mentioned factors (as specified in the *situation* concept).

```
<temporal_situation> ::= BEFORE <situation> | AFTER <situation>
| BETWEEN '(' <situation> ',' <situation> ')'
```

The extensions enable, for example, the definition of *norm 3* that states that players cannot leave the game between its initial and its interruption, as shown below.

```
FORBIDDEN ( player EXECUTE moving:LeaveField ()
                          ( agent.position@pre=inField; agent.position<>inField; )
 BETWEEN ( UTTER(game; si; INFORM(;referee;;[;gameStart;;;;;;])),
           UTTER(game; si; INFORM(;referee;;[;gameStopped;;;;;;])) ))
```

Another norm that makes use of temporal situation is *norm 4*. It states that the referee must send off a player after (s)he receives a second caution in the same match. If player leaves the field of play and (s)he has already been shown a yellow card, the referee must send him(her) off. Note that such *norm 4* is conditioned to the execution of an action governed by *norm 3* and, thus, the *after* condition is exactly *norm 3*.

```
OBLIGED ( UTTER(game;si;CAUTION(;referee;;kicker[;sentOff;;soccerGame;;;;]))
  AFTER ( player EXECUTE moving:LeaveField()
                          ( agent.position@pre=inField;agent.position<>inField; )
          BETWEEN ( UTTER(game; si; INFORM(;referee;;[;gameStart;;;;;;])),
                    UTTER(game; si; INFORM(;referee;;[;gameStopped;;;;;;])) ))
  IF player.yellowCard = true )
```

4.3 Extending the IF Condition

The *if condition* defined in the original normative language is used to introduce conditions over variables, agents' observable attributes or executed dialogical actions. Therefore, by using such language it is not possible to describe *nom 6* since it is conditioned to the execution of a non-dialogical action. Our proposed extension makes possible to specify a condition related to an executed non-dialogical action or to a fired norm.

```
<if_condition> ::= <cond_expression> | NOT '(' <cond_expression> ')'
<cond_expression> ::= <condition> | NOT <condition>
 | <condition> ',' <if_condition> | NOT <condition> ',' <if_condition>
<condition> ::= <action> | <deontic_concept> '(' <action> ')' |...
```

Norm 6 defines that the referee must declare a penalty if a kicker handles the ball. The non-dialogical action of handling the ball is the *if condition* of *norm 6* and can be described as follows.

```
OBLIGED (UTTER(game; si; PENALTY(;referee;kickerTeam;[;penalty;;soccerGame;;;;]))
    IF kicker EXECUTE play:handleBall)
```

4.4 Extending Dialogical Actions

In [8], the authors represent the execution of dialogical actions by the identification of the action (not carried out yet) of submitting an illocution. In their point of view, an illocution is an information that carries a message to be sent by an agent playing a role to another agent playing another role. The *illocution* concept was extended to be possible to omit the agents that send and receive the messages. Not always will be possible to specify the agents that will send and receive the messages while describing the norms. Sometimes only the roles that those agents will be playing can be identified. Moreover, the roles of the sender and receiver can also be omitted. It may be the case that no matter the one is sending a message or no mater the one is receiving it, the norm must be obeyed.

```
<dialogical_action> ::= 'UTTER(' <scene> ';' <state> ';' <illocution> ')'
| 'UTTERED(' <scene> ';' <state> ';' <illocution> ')'
<illocution> ::= <perf>'('<sender>';'<role>';'<receiver> ';'<role>'['<msg>'])'
|...!it is possible to omit the senders, receivers and also their roles
```

Since a message can be sent to several agents, the *receiver* concept was also extended to make possible to describe the group of agents that will be the receiver of the message. Note that it is only possible to describe in the grammar norms that specified messages to be send and not messages to be received. In addition, it is the agent that is receiving the message the one responsible for relating the message being received to a message that should have been sent, in the case of obligations for instance.

```
<sender> ::= <agent>
<receiver> ::= <agent> | <group>
```

By using the extensions provided above for illocution, it is possible to model *norms 1* (Section 4.1), *4* (Section 4.2) and *6* (Section 4.3) that omit the agent identification that is playing the referee role. In such cases, it is not important to identify the agent but only the role that the agent is playing. *Norm 1* also omits the receiver and its role to characterize that the message is being broadcasted. *Norm 4* identifies the role of the receiver but does not identify the agent playing the role since the message to be send does not depend on the agent. Moreover, *norm 6* does not identify the receiver agent but the receiver *team* that will be punished.

4.5 Specifying Messages

The *message* concept has not been specified in the previous version of the normative language. We propose to specify such concept since it may be necessary to provide some characteristics of the messages while describing the norms. The *message* concept was extended according to the parameters defined by an ACL message. While describing *norms 4* and *6* we have used the extended *message* concept to point out the ontology being used to support the interpretation of the content expression.

```
<msg> ::= <conversation_id>';'<contents>';'<language_encoding>';
'<ontology_protocol>';'<reply_by>';'<reply_to>';'<reply_with>';'<in_reply_to>
 |...!it is possible to omit any parameter.
```

5 Implementing Norms

Once we have seen how norms can be described, we need to demonstrate how they are implemented. Similar to the approach presented in [8], we have also used Jess to implement the governance mechanism. Jess is a rule-based system that maintains a collection of facts in its knowledge base. Jess was chosen due two main reasons: (i) it provides interfaces to programs in Java and (ii) it is possible to dynamically change the set of rules defined in Jess from the execution of Java programs. MAS implemented in Java can make use of the knowledge base and the declarative rules provided by Jess. Such MAS can also update the set of rules during the execution.

The use of Jess makes possible to describe facts and rules that are fired according to the stated facts. In our approach, facts are agents' observable attributes, (dialogical and non-dialogical) actions executed by the agents, the norms activated by the rules, and the information about norm violations. The rules are fired according to the executed actions or observable attributes and can activate norms or assert violations.

5.1 The Use of Jess

In Jess, facts are described based on templates that specify the structure of the facts. We have defined a template to define agents' observable attributes and three templates to describe actions: one for describing dialogical actions and two for describing the two different kinds of non-dialogical actions contemplated in the paper (method calling and execution of the action of an agent plan). Besides, we have also described nine templates for describing each of the three norm kinds (obliged, permitted and forbidden) associated with the three different actions (message, method calling and plan execution). In addition, one template was defined for being used to describe norm violations. Such template points out the norm that was violated and the facts that have violated the norm. The two examples below illustrate templates to describe an obligation norm to execute the action of a plan and a violation.

```
(deftemplate OBLIGED-non-dialogical-action-plan
    (slot entity)(slot role)(slot plan) (slot action) (slot attribs (type String))
    (slot contract-pre (type String)) (slot contract-post (type String))
    (slot contract-inv (type String)) (slot beliefUpdated (type String))
    (slot condition (type String)))

(deftemplate VIOLATION (slot norm-violated) (multislot action-done))
```

Rules are composed of two parts. The left-hand side of the rule describes patterns of facts that need to be inserted in the knowledge base in order to fire the rule. The right-hand side defines facts that will be upload to the knowledge base if the rule is fired. In our approach, these facts will be norms or norms' violations. Examples of rules are presented in Sections 5.3, 5.4, 5.5 and 5.6.

5.2 Some Guidelines

For each application norm, there is (usually) a need for describing three rules in Jess. The first rule is used to state the norm by conditioning it to the facts that activate the norm. If the facts are inserted into the knowledge base, the rule is fired and the norm is activated. The second rule deactivates the norm retracting it from the knowledge

base. The period during which some norms are active are limited and conditioned to the addition of some facts in the knowledge base. The third and final rule points out the violations. Prohibitions are violated if facts are inserted into the knowledge base during while they are forbidden and permissions are violated if the facts are inserted into the knowledge outside the period during which they are permitted. The violations of obligations occur if facts are not inserted into the knowledge base in the corresponding period. The following Sections will demonstrate how to implement those rules according to the *temporal situations* and *if conditions* mentioned in Section 4.

5.3 Simple Obligations, Permissions and Prohibitions

Norms that describe obligations, permissions or prohibitions over the execution of actions without defining any temporal situation or if condition are always active. Such norms are never deactivated no matter what happens.

Although it is possible to describe obligations and permissions over the execution of a norm without stating any condition, it is not possible to state violations. For each obligation or permission that is not associated with any temporal situation or if condition, only one rule that states the norm must be described. The obligations characterize that the actions must be executed but do not state when the executions must be checked. Permissions characterize that such actions can always be executed, and, therefore, such norms are never violated by the permitted agents. When permissions are applied to sub-sets of agents, we assume that prohibitions are stated to the ones not permitted to execute the actions. Prohibitions can do be checked and violations can be fired in case the actions are executed. Therefore, for each norm that describes prohibition for the execution of an action, two rules need to be defined: (i) to assert the prohibition; and (ii) to assert the violations if the forbidden facts are added to the knowledge base.

In order to exemplify the use of Jess we describe the implementation of *norm 5*. Rule (i) asserts the prohibition that is not conditioned to any fact. Rule (ii) asserts the violation if a kicker handles the ball.

```
;(rule i)
(defrule forbidden:KickerHandleBall
=> (assert (FORBIDDEN-non-dialogical-action-plan (entity kicker)(plan play)
                                                 (action handleBall))))

;(rule ii)
(defrule violation:KickerHandleBall
?fact <-(non-dialogical-action-plan (entity kicker)(plan play)(action handleBall))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (entity kicker)(plan play)
                                                    (action handleBall))
=> (assert (VIOLATION (norm-violated (fact-id ?forbidden))
                      (action-done (fact-id ?fact)))))
```

5.4 Norms Regulating Actions Executed Before the Occurrence of a Fact

Obligations for executing an action X before the occurrence of a fact W are verified testing if X has been executed before W occurs. For governing such norms three rules are defined: rule (i) asserts the obligation for execute X; rule (ii) retracts the obligation if X has been executed and W occurs; and rule (iii) asserts a violation if W occurs but X has not been executed (what can be verified by the existence of the obligation).

Permissions for executing an action X before the occurrence of W are verified testing if X is executed after W. In such case, the execution of X is not permitted. These norms are governed by three rules: rule (i) asserts the permission for execute X; rule (ii) retracts the permission if W occurs; and rule (iii) asserts a violation if W occurs and X is executed.

Prohibitions for executing an action X before the occurrence of an action W are verified testing if X is executed and W has not occurred. Such norms are also governed by three rules: rule (i) asserts the prohibition; rule (ii) retracts the prohibition if W occurs; and rule (iii) asserts a violation if X is executed and W has not occurred (what can be verified by the existence of the prohibition). We assume that W can occur many times but obligations should be fulfilled before the first time it occurs and permissions and prohibitions are only active before its first occurrence.

Norm 1 is a good example to illustrate the implementation of norms that govern the actions that must be executed before another one. Since the norm defines that a referee is *obliged* to check the equipment of the players *before* starting the game, three rules was defined to govern such norm. Rule (i) states the obligation. Rule (ii) retracts the obligation if the referee has checked the player equipment when the game starts. Rule (iii) asserts a violation if the game has been started and the obligation still holds informing that the referee has not checked the equipment. The obligation governs a non-dialogical action that must be executed after a dialogical action.

```
;(rule i)
(defrule obliged:CheckEquipment
 =>(assert (OBLIGED-non-dialogical-action-plan (entity referee)(plan managingGame)
     (action checkEquipment)(attribs players)
     (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))))

;(rule ii)
(defrule retract:CheckEquipment
(non-dialogical-action-plan (entity referee)(plan managingGame)
                            (action checkEquipment)(attribs players))
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
?obliged <- (OBLIGED-non-dialogical-action-plan (ntity referee)
       (plan managingGame)(action checkEquipment)(attribs players)
       (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))
=> (retract ?obliged))

;(rule iii)
(defrule violation:CheckEquipment
?fact <- (dialogical-action (scene game)(state si)(performative inform)
                            (sRole referee)(message "gameStart"))
?obliged <- (OBLIGED-non-dialogical-action-plan (ntity referee)
       (plan managingGame)(action checkEquipment)(attribs players)
       (condition "BEFORE UTTER(game; si;INFORM(;referee;; [;gameStart;;;;;;]))"))
=> (assert (VIOLATION (norm-violated (fact-id ?obliged))
                   (action-done (fact-id ?fact)))))
```

5.5 Norms Regulating Actions Executed After the Occurrence of a Fact

Obligations for executing an action X after the occurrence of Y (or if Y occurs) cannot be governed since it is not possible to affirm that the execution of X will never occur after the execution of Y. It is not possible to state a rule that fires a violation for such norm since the action X can be executed anytime after Y has occurred. In order to govern such norms it is necessary to state any temporal situation limiting the time

for the execution of X after Y has occurred. The temporal concept *between* should be used instead of *after* or *if* for governing such obligations. *Norms 4* and *6* are examples of norms that should be implemented by using *between*, as depicted in Section 5.6.

Permissions for executing X after the occurrence of Y can be governed by two rules: rule (i) asserts the permission if Y occurs; and rule (ii) asserts a violation if X is executed but Y has not occurred yet (i.e., there is no permission for execute X).

The governance of prohibitions for executing X after the occurrence of Y is the opposite to the governance of the related permission. Such governance is also characterized by two rules: rule (i) asserts the prohibition if Y occurs; and rule (ii) asserts a violation if X is executed after Y has occurred or if Y is true.

In order to exemplify a norm that use the *if condition* we refer to *norm 2*. This norm defines that the coach cannot execute an action that substitutes players if the number of substitutions is equal or greater than 3. The prohibition governs a non-dialogical action that is condition to the state of an object.

```
;(rule i)
(defrule forbidden:PlayerSubstitution
(attribute-value (objectORagent team)(attribute substitutions)(value 3))
=> (assert (FORBIDDEN-non-dialogical-action-plan (role coach)(plan managingTeam)
               (action substitutePlayer)(attribs outPlayer,inPlayer,team)
               (contract-pre "team.coach=coach")
               (contract-post "team.substitutions=team.substitutions@pre+1 AND
                               team.playersInField->excludes(outPlayer) AND
                               team.playersInField->includes(inPlayer)") )))

;(rule ii)
(defrule violation:PlayerSubstitution
?fact1 <- (non-dialogical-action-plan (role coach)(plan managingTeam)
                                      (action substitutePlayer))
?fact2 <- (attribute-value (objectORagent team)(attribute substitutions))
?forbidden <-(FORBIDDEN-non-dialogical-action-plan (role coach)(plan managingTeam)
                (action substitutePlayer)(attribs outPlayer,inPlayer,team)
                (contract-pre "team.coach=coach")
                (contract-post "team.substitutions = team.substitutions@pre+1 AND
                                team.playersInField->excludes(outPlayer) AND
                                team.playersInField->includes(inPlayer)"))
=> (if (>= (fact-slot-value ?fact 2) 3 ) then
       (assert (VIOLATION (action-done ?fact1  ?fact2)
                          (norm-violated ?forbidden))) ))
```

5.6 Norms Regulating Actions Executed Between the Occurrence of Two Facts

A norm that states an obligation for executing an action X after the occurrence of Y and before the execution of W is governed by three rules: rule (i) asserts the obligation for execute X if Y occurs; rule (ii) retracts the obligation if X is executed and if W occurs; and rule (iii) asserts a violation if W occurs but X has not been executed.

The permission for executing X between the occurrence of Y and W is governed by the following four rules: rule (i) asserts the permission for execute X if Y occurs; rule (ii) retracts the permission if W occurs; rule (iii) asserts a violation if W occurs and X is executed; and rule (iv) asserts a violation if X is executed but Y has not occurred yet (i.e., if the permission for executing X has not been fired yet).

Prohibitions for executing X between the occurrence of Y and W are governed by three rules: rule (i) asserts the prohibition if Y occurs; rule (ii) retracts the prohibition if W occurs; and rule (iii) asserts a violation if X is executed, Y has occurred but W

has not occurred, i.e., X is executed and the prohibitions is still activated. Note that the rules that govern both prohibitions and permissions while using the temporal concept *between* are the combination of the rules used to govern such norms using the *after* and *before* temporal concepts.

The use of *between* can be exemplified by *norm 3*. It states that the player is forbidden to leave the field between the beginning and the end of the game. The norm defines a prohibition to execute a non-dialogical action limited by the execution of two dialogical actions. Rule (i) asserts the prohibition if the first dialogical action is executed, rule (ii) retracts the prohibition if the second dialogical action is executed and rule (iii) declares a violation if the non-dialogical action is executed during while it is being prohibited.

```
;(rule i)
(defrule forbidden:LeaveField
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
 => (assert (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
           (action leaveField)(contract-pre agent.position@pre=inField)
           (contract-post agent.position!=inField ))))

;(rule ii)
(defrule retract:LeaveField
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStop"))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
                (action leaveField)(contract-pre agent.position@pre=inField)
                (contract-post agent.position!=inField ))
 => (retract ?forbidden))

;(rule iii)
(defrule violation:LeaveField
(dialogical-action (scene game)(state si)(performative inform)(sRole referee)
                   (message "gameStart"))
?forbidden <- (FORBIDDEN-non-dialogical-action-plan (role player)(plan moving)
                 (action leaveField)(contract-pre agent.position@pre=inField)
                 (contract-post agent.position!=inField ))
?fact <- (non-dialogical-action-plan (role player)(plan moving)(action leaveField)
            (contract-pre agent.position@pre=inField)
            (contract-post agent.position!=inField ))
=> (assert (VIOLATION (norm-violated (fact-id ?forbidden))
                   (action-done (fact-id ?fact)))))
```

Sections 5.3 and 5.5 point out that some obligations over the execution of a norm that cannot be governed. Since obligations need not to be fulfilled immediately after they were declared, it is necessary to inform the period during which the agents are being obligated to execute the action in order to govern them. *Norms 6* and *4* are very good examples of such obligations. *Norm 6*, for instance, defines that the referee must declare a penalty if a kicker handles the ball. However, this norm does not define how much time the referee has to fulfill its obligation. Therefore, it is not possible to affirm that the obligation was not fulfilled since it can be at any time. In order to properly regulate such norm it is needed to provide a limit till when this obligation must be fulfilled. *Norms 6* was adapted to inform that the referee has 1 minute to declare the penalty after the kicker has handled the ball.

```
OBLIGED ( UTTER(game; si; PENALTY(;referee;kickerTeam;[;penalty;;soccerGame;;;;]))
   BETWEEN ( kicker EXECUTE play:handleBall, 1 MINUTES OF kicker EXECUTE
             play:handleBall ) )
```

6 Conclusion

This paper proposes the implementation of norms[4] that govern dialogical and non-dialogical actions by using Jess. The governance system proposed in [6] receives (not always true) testimonies about executed actions that are related to norm violations. After judging the testimonies and concluding that the actions really were executed, such information is uploaded to the Jess knowledge-based. The set of Jess rules are, then, checked and the related norms and violations are fired. The fired norm or violation is also facts accumulated in the Jess database. We have implemented in Jess at least one norm taking into account the three *deontic* concepts, the proposed temporal situations and if conditions presented in the paper by using the soccer game.

Note that the Jess system only receives one information about the execution of an action at a time. Independently of the order of the execution of the actions, the first information sent to Jess is the one that will be processed. If two actions are executed at the same time, the first information to achieve the Jess system will be processed.

Although the current version does not contemplate sanctions and awards, it can be easily extended in order to do so. The sanctions should be provided when the related violations are fired. The awards should be supplied when the norms are retracted and no violation of such norms has been fired. In addition, a (semi)automatic approach for generating Jess rules according to the norms specified by the use of the normative language could be developed.

An automatic approach for generating Jess rules from the norms specified by the use of the normative language is being developed. Our intention is to use such transformer during design time to automatically generate the rules for the specified norms and also during runtime. In case the agents are able to specify new norms according to the normative language during runtime, they could use the proposed transformer to automatically generate new rules and publish them in the Jess engine. We are also investigating the possibility of modifying one of the already available rules. Such transformation should be based on the guidelines provided in section 5.2 and also on its specialization provided in the following sub-sections.

References

1. Boella, G., van der Torre, L.: Regulative and Constitutive Norms in Normative Multi-Agent Systems. In: Proceeding of KS, pp. 255–265. AAAI Press, Menlo Park (2004)
2. Artikis, A., Kamara, L., Pitt, J., Sergot, M.: A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. In: Leite, J.A., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS (LNAI), vol. 3476, pp. 221–238. Springer, Heidelberg (2005)
3. Broersen, J., Dignum, F., Dignum, V., Meyer, J.: Designing a deontic logic of deadlines. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 43–56. Springer, Heidelberg (2004)
4. Cranefield, S.: A Rule Language for Modelling and Monitoring Social Expectations in Multi-Agent Systems. In: Boissier, O., Padget, J.A., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM 2005 and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 246–258. Springer, Heidelberg (2006)

[4] The full normative language described in the paper and the Jess program used to illustrate our approach are available at http://maude.sip.ucm.es/~viviane/products.html

5. Dignum, F., Broersen, J., Dignum, V., Meyer, J.: Meeting the deadline: Why, when and how. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) FAABS 2004. LNCS (LNAI), vol. 3228, pp. 30–40. Springer, Heidelberg (2004)
6. Duran, F., Silva, V., Lucena, C.: Using Testimonies to Enforce the Behavior of Agents. In: Sichman, J.S., et al. (eds.) COIN 2007 Workshops. LNCS (LNAI), vol. 4870, pp. 232–244. Springer, Heidelberg (2008)
7. García-Camino, A., Rodríguez-A, J., Sierra, C., Vasconcelos, W.: Norm-Oriented Programming of Electronic Institutions. In: Proceedings of AAMAS, pp. 670–672. ACM Press, New York (2006)
8. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: Implementing Norms in Electronic Institutions. In: Proceedings of AAMAS, pp. 667–673. ACM Press, New York (2005)
9. López, F.: Social Power and Norms: Impact on agent behavior. PhD thesis, Univ. of Southampton (2003)
10. López, F., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: Proceedings of AAMAS, pp. 674–681. ACM Press, New York (2002)
11. Meyer, B.: Object-Oriented Software Construction Prentice Hall, 2nd edn (1997)
12. Singh, M.: An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. In: Artificial Intelligence and Law, vol. 7(1), pp. 97–113. Springer, Heidelberg (1999)
13. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Implementing Norms in Multiagent Systems. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 313–327. Springer, Heidelberg (2004)