Condition-Based Knowledge Representational Structure for Identifying Norms Violation In Logic-Based Normative Systems

BABALOLA Moyin Florence and AKINKUNMI Opeoluwa Babatunde

ABSTRACT

Normative systems use norms to regulate agents' activities in a Multi-Agent System (MAS). Existing methods infer the violation of a norm as the failure of an intelligent software agent to guarantee the effect of the norm at a certain time (fixed). This type of inference is without recourse to the norm's condition and its identity that warranted the intelligent software agent failure. This study was therefore designed to develop Condition-based Knowledge Representational Structures (CKRS), capable of making violation inferences about the condition and the identity of the violated norm for open agent societies. Norms were formalised in a Horn's Clause (HC). The formalised norms were assigned norm identifications (Id) that they pass on to the normative tokens inferred from them. The language was applied to real life norms arising from lecturing, registration and examination domains using data from University of Ibadan, Nigeria. A reified Temporal Constraint Structure (TCS) that combined qualitative and quantitative relationships between actions and situations was used to describe the time constraints between situations (s) and the action types (a-type) in each norm. Satisfaction Rules (SRs) that determined if temporal constraints (tc) hold for actions (act) and situations were developed. Norm validity was represented as a relation between a norm's identity and the time of its validity. The CKRS was compared to existing methods in order to ascertain its effectiveness in identifying norms violation in MAS.

Keywords: Multi-Agent Systems, Horn's Clause, Temporal Constraints Structure, Condition-Based Knowledge Representational Structures, Satisfaction Rules, Norms, Normative Token.

INTRODUCTION

Normative systems use norms to regulate agents' activities in a Multi-Agent System (MAS). In other words, a normative system is a multi-agent system (MAS) associated with a set of norms (social law) (Artikis, 2003; Artikis *et al.*, 2009). Several works have been carried out in MAS on how norms can be used as mechanisms to coordinate the behavior of the intelligence software agents by describing the actions that are *obligatory*, *permitted* and *prohibited* for the agents. Specifically, norms are used in MAS to cope with autonomy, different beliefs, interests and desires of the agents that cohabit in the system. Based on the normative descriptions and the actual (past and present) actions of the agents, the system should detect the deviating behavior of any agent in the system. Artikis and Sergot, (2010), specified norm-governed computational societies using two action languages; the *C*+ language and the Event Calculus (Kowalski and Sergot, 1986), and were executed using the Causal Calculator Software and the Society Visualise Software respectively in order to predict the future. Existing normative systems infer the violation of a norm by detecting the failure of an agent to guarantee the effect of the norm at a certain time (fixed). This type of inference is without recourse to the norm's condition (agent, action type, norm situation, flexible time) and its identity that warranted the intelligent software agent failure. Furthermore, it is pertinent to determine the validity of the violated norm(s) as the time the violation took place. A norm is said to be valid at a point in time if it is part (member) of the system in consideration at that point in time and the interval during which the norm is valid, is called the *external time* of that norm (Marin and Sartor, 1999; Royakkers and Dignum, 1997). The aim of this research is to develop Condition-based Knowledge Representational Structures (CKRS), capable of making violation inferences about the condition and the identity of the violated norm.

RELATED WORKS

In the literature, several authors such as (Stratulat, *et al.*, 1999; Marin and Sartor's, 1999; Castelfranchi *et al.* 2000; Stratulat, *et al.*, 2001a, 2001b; L'opez y L'opez, 2003; Artikis, 2003; Sadri *et al.*, 2006; Artikis *et al.*, 2009; Artikis and Sergot, 2010; Ahmad *et al.*, 2011, among others) have formalised norms in Artificial Intelligence and Law. Jones and Sergot (1993), were able to formalised the deontic status and action of an agent in a system using modal deontic logic, they did not consider norm condition (situation) and effect, as well as the temporal constraints relating to the norm. Mostly because deontic logic can not explicitly represent time, it makes it difficult for them to use to represent real life norms. This is one reason that lead to using First order logic (Reified) in this research. In Sadri *et al.* (2006) work, the idea of reifying time constraints was used however, their constraints were between time points, which will not be able to capture both qualitative and quantitative constraints between indefinite times. This gap is bridged in this study by representing the time interval constraints using time point images (TPI), so that indefinite times can be represented. It was also observed that though Artikis *et al.*, (2009) and Artikis and Sergot (2010) formalised sanction but they avoided the issue of temporal constraints between norm's condition and their effects, unlike formalisms presented in this study. On Stratulat *et al.* (1999, 2001a, 2001b) works, it was discovered that they based their detection of violations of normative positions as being the non-execution of an obligatory action or the execution of a forbidden action. Stratulat *et al.*, (200la) proposed a formalisation of the life-cycle of normative position using first order logic to address the issues of norm *violation* and *validity* of the norms. Their violation was represented with the fluent *V(ag, act, int*): which means agent *ag* violated a normative position with respect to the execution of an action of type *act* over the interval *int*. A key aspect of their definition of a violation is the time of observation of the violation. An example of their violation is:

$$\begin{aligned}&\text{holds}\ (t, V\ (\text{agent}, \alpha, [t_1, t_2]))\ \equiv\\ &\qquad \text{holds}\ (t; O\ (\text{agent}, \alpha, [t_1; t_2])) \wedge [t_1, t_2] < [t, \infty] \wedge\\ &\qquad\qquad \forall\ \text{act}\ (\text{instance\_of}\ (\text{act}, \alpha) \Rightarrow\\ &\qquad\qquad\qquad \text{holds}\ (t, \text{failed}\ (\text{agent}, \text{act}, [t_1, t_2])))\end{aligned}$$

In this statement, *t* is the time when the violation is observed or detected. This type of inference is without recourse to the norm's condition (agent, action type, norm situation(condition), flexible time) that warranted the intelligent software agent failure and identity of the violated norm. This makes an agent (the normative agent system) lose sight of the conditions under which a violation took place. Take for instance a real life scenario; in an electronic institution where there is a norm which says agent has an obligation to supply an order within forty eight (48) hours after receiving the placement order and the acknowledgment of payment for the order. Generally, there is always a condition warranting an action to take place in a dynamic world. In the e-institution scenario given here, the conditions required the agent to oblige (or fail to oblige) to the norm are:

i) The agent receiving the placement order and,
ii) Receiving the acknowledgment of payment for the order.

The ignored information about the violation makes it difficult for an agent receiving it to know why a violation has taken place and perhaps learn how to avoid such a violation the next time, this is main gap paper aimed to solve. Therefore, to completely formalise norms in a dynamic normative agents system (NAS), there is an absolute need to base the formalisation on the norm's situation (condition) as well as action (or action type).

Apart from the above, the issues of time interval (*internal time*) and *validity* of norms remains pertinent in normative agent systems (NAS) formalisation. Internal time of norms refers to the time interval in which the condition of the norm should hold in order to produce the norm's effect (Artikis, 2003; Marin and Sartor's, 1999). However, the temporal constraints reasoning for NAS formalisation should be flexible enough to capture dynamically the temporal constraints between norms conditions and effect, not just the temporal constraints based on fixed time as in (Stratulat *et al.*, 200la; and Sadri, *et al.*, 2006). In other words, it is very important in formalizing the life-cycle of normative positions in a dynamic normative agents system (NAS), to make use of a reified Temporal Constraint Structure (TCS) which described constraints between the times of situations and the action types they warrant in each norm in a dynamic way.

Furthermore, it is equally important in deciding if an agent violates a particular norm in NAS, the violation rule should not be based on the existence or otherwise of an action and temporal constraints only within a time interval, rather should be formalised in terms of the existence or otherwise of an action type, in response to the (or anticipation of) situation (condition) satisfying the required temporal constraint and the *validity* of the norm at the time of the given situation (condition). A norm is considered *valid* at a point in time if it is part (member) of the legal system in consideration at that point in time. The validity interval of a norm is the interval during which the norm is valid (that is, *external time*).

3 THE CONDITION-BASED KNOWLEDGE REPRESENTATIONAL STRUCTURES (CKRS)

Norms are viewed as the need for an agent to carry out or avoid carrying out an action in response to (or in anticipation of) a condition (here represented by situations like in Situation Semantics (Schubert, 2000)) within some time constraints. Thus the norms in this study have the following components:

i) Normative position (e.g. Obligation, Prohibition, or Permission)
ii) The agent itself,
iii) An action type,
iv) A situation,
v) The time constraint between the action and situation and
vi) The norm's identity.

A normative position is treated as the relation (or predicates) bringing the other five elements together. The signature of the language is subsequently defined.

3.1 THE LANGUAGE: SORTS, FUNCTIONS AND PREDICATES

The language of the designed normative system in this paper is a reified many sorted first order logic with equality. The main sorts are *Agents*(Ag), *Action-types* (ActT), *Action* (Act), *Fluents* (Fl), *Events* (Ev), *Processes* (Pr), *Situations* (Sit), *Time-Intervals*(lnt), *Temporal-Constraints* (TC), *Norm-identifications* (Nld). The Domain Sorts include entity classes in the domain such as *Class Classrooms*, Objects. There are three relations used to assert the fact that an agent is in some normative position. These are Obligations, Prohibitions and Permissions. Each of this is a quaternary relation. The signatures are stated below:

Obligation: Ag x ActT x Sit x TC x Nld → Boolean
Prohibition: Ag x ActT x Sit x TC x Nld → Boolean
Permission: Ag x ActT x Sit x TC x Nld → Boolean

An obligation is true when an agent is expected to take a particular action of a certain type when a situation arises, in such a way that the times of agent's action and the situation satisfy the temporal constraint. Similarly, a permission (prohibition) is true when an agent is (not) allowed to carry out any action of a certain type.

There are also predicates that denote Allen's (1984) interval relations and the other predicates built on them.
Meets, After, Starts, Finishes, Contains, Overlaps: Int x Int → Boolean
There are other temporal interval relations which definition is based on those of Allen. Those include the following:

Within, Subinterval, Disjoint: Int x Int → Boolean

The Holds predicates are used to associate fluents that are true with the situations in which they are true. Apart from this, Actions and Action types are related by an instance relation.

Holds: F1 x Sit → Boolean
Holds_in: F1 x Sit → Boolean

The following predicates describe relations involving actions. The first one ActionType describes the relation between an action instance and the action type, while Actor is a relation between an action instance and the agent that carries it out.

ActionType: Act x ActT → Boolean
Actor: Agent x Act → Boolean

With respect to situations there are two major relations signified by the predicates SubSit and SubSitT respectively denote sub-situation and temporal sub situation relations.

SubSit: Sit x Sit → Boolean
SubSitT : Sit x Sit → Boolean

A pair of situations is a member of the sub-situation (SubSit) relation if every fluent that holds in the first situation also holds in the second situation and their time are the same. A pair of situations is a member of the temporal sub-situation relation (SubSitT) if the time of the first situation is a subinterval of the second.

Another relation denoted by the predicate Satisfy relates two time intervals and a temporal constraint they satisfy:

Satisfy: Int x Int x TC → Boolean

ISSN 2229-5518

Satisfy is true when two time intervals satisfy the temporal constraint. Another relation is between a norm and its time of validity. The signature of the predicate is:

Valid: Norm-Id x Int $\rightarrow$ Boolean

There are also some group of functions. One group of functions of 0 or 1-arity return actions.

arrive: Location $\rightarrow$ Act

The other group of 1-arity and 2-arity functions returns fluents (A fluent is a reified propositional description of a partial state of the world). A good example of this is when a fluent is defined by the occurring of an event. Another function/prog is when a fluent is defined by a process being in progress,

occurring: Ev $\rightarrow$ F1

prog: Pro $\rightarrow$ Fl

The other examples include an agent possessing an object and the fact that an event is happening at a location.

possess: Ag x Object $\rightarrow$ F1

venue: Event x Location $\rightarrow$ F1

There is a maintain function which returns the kind of action that an agent takes when he keeps a fluent holding in a situation:

maintain: F1 $\rightarrow$ ActT

Apart from these there are four time functions TimeA and TimeS, TimeE and TimeP which map actions and situations, events and processes respectively to time.

timeA: Act $\rightarrow$ Int

timeS: Sit $\rightarrow$ Int

timeE: Ev $\rightarrow$ Int

timeP: Pr $\rightarrow$ Int

A Time point image (TPI) may be either of the constants (B, E} or an application of the time displacement function tdisp to either B or E and an integer. The signatures are given as follows:

tdisp: TPI x Integer $\rightarrow$ TPI

A basic time constraint is obtained by the application of one of the functions *eq* (which means equal to), *le* (which means less than or equal to), *lt* (which means less than), *gt* (greater than) and *ge* (greater than or equal to).

eq, *le*, *It*, *gt*, *ge* : TPI x TPI $\rightarrow$ TC

Other temporal constraints can be obtained by applying Boolean functions on existing temporal constraints.

and: TC x TC $\rightarrow$ TC

of: TC x TC $\rightarrow$ TC

neg: TC $\rightarrow$ TC

3.2 SEMANTICS OF THE LANGUAGE

This subsection focuses on describing the logical entities of the representation such as; situations, fluent, action type, temporal constraints structure (TCS), satisfaction rules and their interactions.

3.2.1 SITUATIONS

The notion of situations as used in this thesis covers such concepts as states and the occurring of events and processes, A common name for all these in the literature is the term eventualities as used by Galton, (2005). However, the term situations are used because the usage of the term is akin to the Situational Calculus and the domain of situations and their relations are similar to those used by the situation semantics inspired work of Schubert (2000). Like in Schubert's FOL**, a fluent may either partially or fully characterize a situation. However unlike FOL**, the possibility of ever associating complex (logical) sentences with situations is ruled out. In fact the fluent is treated as the range for a function.

The predicate Holds is used to associate a fluent with a situation that it fully characterizes, while the predicate Holds_in is used in the case of characterizes. Holds_in is a generalization of Holds. That relationship between them is formalised thus:

$\forall$f, s. Holds(f, s) $\rightarrow$ Holds_in(f, s)

In semantic terms, a situation is treated (in this paper) as a solid entity so that it can only be true over a specific interval.

Sometimes the occurring of an event may fully characterize a situation but the fact that an agent plays a specified role in the event only partially characterizes the same situation. For instance, if an event of the type class holds in a certain situation, then some particular agent must play the role of a teacher in that situation, while a number of other agents must play the role of students. This can be expressed:

$\forall$s. Holds(*occurring*(class), s) $\wedge$ Event_type(class, Class) $\rightarrow$

ISSN 2229-5518

$\exists!$ a. Holds_in(*role*(a, Teacher), s)

$\forall$s. Holds(*occurring*(class), s) $\wedge$ Event_type(class, Class) $\rightarrow$

$\exists$a. Holds_in(*role*(a, Student), s)

These kind of partial characterizations in terms of Holds_in are important for us to be able to specify norms of conduct when an event brings together different agents. In general if a fluent partially characterizes a situation then there is a situation it fully characterizes which is a sub-situation of the original situation. This is formalised as:

$\forall$f, s. Holds_in(f, s) $\leftrightarrow$ $\exists$ s_1 Holds(f, s_1) $\wedge$ SubSit(s_1, s)

$\forall$s, s_1. SubSit(s, s_1) $\leftrightarrow$ ($\forall$f. Holds_in(f, s) $\rightarrow$ Holds_in(f, s_1)) $\wedge$ timeS(s) = *timeS*(s_1)

Another relation between fluent and situation is denoted by the predicate Holds_within.

$\forall$f, s. Holds_within(f, s) $\leftrightarrow$ $\exists$ s_1 Holds(f, s_1) $\wedge$ SubSitT(s_1, s)

$\forall$s, s_1. SubSitT(s, s_1) $\leftrightarrow$ Subinterval(*timeS*(s), *times*(s_1))

The time of an event which occurring constitutes a situation is the same as the time of the situation. That is formalised thus:

$\forall$e, s. Holds(*occurring*(e), s) $\rightarrow$ *timeE*(e) = *timeS*(s)

3.2.2 FLUENTS AND ACTION TYPE

In this language, action types and fluent are reified. In principle reification is about making something which ordinarily would be regarded as a proposition into a thing that can be quantified and reasoned about (Galton 2005). In Artificial Intelligence (AI) literature eventualities such as states, events and processes which are normally represented as proposition are reified.

In reifying, the predicates involved in the proposition are given the status of functions of appropriate arity returning the appropriate kind of eventuality such as fluent or action type. As pointed out earlier, the occurring of an event and the progress of an event can be

regarded as fluents. Either of this kind of fluents can completely characterize a situation. For example, the situation so is fully characterized by the occurring of an event e:

Holds(*occurring*(e), s_0)

Similarly the situation s_i is fully characterized as a process p being in progress:

Holds(*prog*(p), s_1)

3.2.3 TEMPORAL CONSTRAINTS STRUCTURE (TCS)

Temporal constraints are needed to specify the constraint between the time of a norms effect and the situation. Time constraints were reified as done by Sadri *et al.*, (2006). However, the difference between this approach and Sadri *et al.* is that their constraints involve actual times while this involves time point images (TPIs). The time point images are used to model such constraints as *within ten minutes of the beginning of the class and not later than five minutes to the end of the examination*.etc.

For example, Sadri *et al.* (2006) gave an example of formalizing the prohibiting any agent from parking at the city centre between 10 Hours and 17 Hours as:

prohibited(act(park, a, city centre), T, 10 < T < 17)

However, the reality is that the times given for this constraint are given relative to the start of any new day. In that sense the times used in this constraint are really not absolute times. Another example that illustrates this from Stratulat *et al.* (200la, 2001b) is the example of an obligation to pay taxes between January 1 and 3 1 every year on the part of an agent a. That is rendered as a function O returning a fluent may hold at a certain time. This fluent is rendered as:

O(a, pay-taxes, [January - 1 , January-31])

Again these times are erroneously taken as time points. However, they are at best images which are relative to the beginning of a year. They only become a time point when those dates are located within a specific year, just as the time point in the last example needs to be situated within a specific day for it to be an actual time point.

The approach in this study on the norm representation, therefore, is to relate the time of the condition of a norm (which was expressed as a situation), with that of the effect of the norm which is the time of the action of the type specified as a response to the situation. A basic temporal constraint is composed by the application of one the functions *eq* (equal), *ge* (greater than or equal to), *le* (less than or equal to), *gt* (greater than) and *lt* (less than) to an ordered pair of TPI, the first relating to the action (or norm's effect) while the second is related to the situation (norm's condition). For example:

eq(B, E) means the action begins when the warranting situation ends.

eq(*tplus*(B, 3), E) means that the action is to start exactly 3 time units after the situation ends.

Other temporal constraint can be composed from other temporal constraints by the functions and, or as well as neg. For example, *and*(*eq*(B, B), *eq*(E, E)) describes an action that must take place at exactly the same time as the situation.

Each of Allen's qualitative relations (Allen, 1984) can be represented by the proposed TCS. The following are the equivalences between Allen's interval relations and the proposed TCS.

Before = *lt*(E, B)
Overlaps = *and*(lt(B, B), *lt*(E,E))
Contains = *and*(*lt*(B, B), *gt*(E, E))
Starts = and(*eq*(B, B), *lt*(E, E))
Finishes = *and*(*gt*(B, B), *eq*(E, E))
Meets = *eq*(E,B)

A major advantage of the TCS representation is that it can represent constraints that combine both qualitative and quantitative relationships. Examples of such relationships are:

Start not later than 4 units of time into = *ge*(B, *tplus*(B, 4))
End 10 units of time into = *eq*(E, *tplus*(B, 10)

These are the kinds of constraints that norms may contain as holding between norm conditions and their effects. The following section presented a formalisation of the real life norms.

3.3 REPRESENTATION OF NORMS

A norm is basically a rule. Each of these *normative rules* help in making inferences about what an agent is expected to do or desist from doing within a certain time frame. The inferences made from these normative rules are referred to as *normative tokens*.

An obligation requires an agent to carry out an action of a certain type within a time constraint if it is playing a certain role within a certain context. An obligation is violated by an agent if when it finds itself playing the specified role within such a context it fails to carry out an instance of that action type within the time constraint. The structure of the predicate for representing obligations is presented thus:

Obligation(Agent, Action-type, Situation, Temporal-Constraint, Norm-Id)

Norm-id is an identifier for the norm rule that produced the actual normative token which is an obligation. Every obligation that is produced by the norm bears the same norm-id. As such every rule that helps to infer obligations, prohibitions and permissions has a unique norm-id that it carries. This kind of rule naming is refered to by the term *rule reification* which is similar in spirit to the notion of Davidson's reification (Galton 1991), This is illustrated with Norm 3.1 and 3.2 as examples of rules generating this kind of normative tokens.

Norm 3.1

A teacher assigned to teach a class must arrive either on time or not later than 10 minutes into the time of a class.

∀a, v, s,

Obligation(a, *arrive-at*(v), s, *and(le*(E, *tplus*(B,10)), *ge*(E, B)), OB101) if

∃ e.
Holds(*occurring*(e), s) ∧
EventType(e, Class) ∧
Holds-in(venue(e, v), s) ∧
Holds-in(role(a, Teach), s))

Norm 3,2

Student must register for his/her courses in a semester within one month of the commencement of the semester.

∀a, o, s.

Obligation (a, *register*(a,sem)), s, *and(and*(*ge*(E, B), *le*(E,B, 30)), *ge*(B,B)), OB102) if

$\exists s_1, s_2$
Holds(*studentship*(a),S_1) ∧
ProcessType(Sem, Semester) ∧
Holds(*prog*(sem), s) ∧
Within(*timeS*(s), *timeS*(s_1))∧
¬ OnSuspension(a, sem)

A prohibition is a norm that disallows an agent to carry out an action of a certain type within some time interval that has a temporal relation with the warranting situation (condition) of the prohibition. The structure of a prohibition is given thus:

Prohibition(Agent, Action-type, Situation, Temporal-Constraint, Norm-id)

The norm 3.3 and norm 3.4 illustrate examples of rules that infer such normative tokens.

Norm 3.3

Student must not be allowed to come in for an examination thirty (30) minutes after the commencement of the examination.

∀a, s, e.

Prohibition(a, *arrive*(v), s, *gt*(E, *tplus*(30)}, PR0103), if
Holds(*occuring*(e), s) ∧
EventType(e, Examination)) ∧
Holds(Venue(e,v), s) ∧
Holds-in(*role*(a, Candidate), s)

Norm 3.4

It is prohibited for members of university community to release confidential document of the university to public domain without authorization.

∀a, doc, s,

Prohibition(a, *release*(doc), s, *or*(*and*(*lt*(B, E), *ge*(B, B)), *and*(*le*(E, E), *gt*(E, B))), PRO014) if
∃s_1,s_2, u.
Holds(*alive*(doc), s_1) ∧
Holds(Statusdoc(doc, Confidential) ∧
Holds(employ(u, a), s_2) ∧
University(u) ∧
Submterval(*timeS*(s), *timeS*(s_1)) ∧
Submterval(*timeS*(s), *timeS*(s_2)) ∧

A permission is a norm that allows an action by an agent as a result of a certain situation arising. The structure of permission is similar to that of the other kinds of norms as seen below:

Permission(Agent, Action-type, Situation, Temporal-Constraint, Norm-id)

Examples of this is given in norm 3.5 and norm 3.6

Norm 3.5

Members of the university community are permitted to put on their official identity card while on duty.

∀a, s,

Permission(a, *put_id-on*(a), s, *and*(*ge*(B, B), *le*(E, E)), PER010) if
∃u, s_1
Holds(*onDuty*(a), s)
Holds(employ(a, u) s_1)
University(u)
Within(*timeS*(s), *timeS*(s_1)

Norm 3.6

Lecturer is permitted to give reading books on his assigned course to the student at the beginning of lecture in a semester.

∀a, b, s.

Permission(a, *give*(b), s, *le*(E, B), PER012) if
∃co.
Holds(*prog*(co), s) ∧
ProcessType(co, Course) ∧
Holds_in(role(a, Teacher), s) ∧
Holds_in(role(b, ReadingBook), s)

Having represented norms, it is important to discuss the representation of the validity of those norms. Norm validity is a relation between a norm and the interval over which the norm is valid. For example, the legal norm from the formalisation of the British Nationality Act 1981 (Segot *et al*., 1986).

The norm *any person born in the UK* becomes a British citizen is a norm that was only valid until 1981. A representation of that norm in the proposed language is presented thus:

∀x, s

Obligation (HMG, *grant-citizenship*(x), s, *eq*(E, B), NBB-1) if
Holds(born-in(x, UK), s)

However, any agent interpreting that norm on behalf of Her Majesty's Government (HMG) must be aware that the validity of the norm is from 1950 and 1981.

Validity(NBB-l,(1950, 1981))

As such it is only if the norm situation s happened within that interval of validity that the norm NBB-1 is valid. Validity is an important condition for deciding norm violation as we shall demonstrate.

3.4 SATISFACTION RULES

The proposed Satisfaction Rules (SRs) are used to decide whether or not a particular pair of time intervals satisfies a temporal constraint. Those time intervals may be times of actions, situations or even events. Some of the SRs are recursive such as:

∀act,s,tcl,tc2

Satisfy-Cons(*timeA*(act), *timeS*(s), *and*(tcl, tc2)) if
Satisfy-Cons(*timeA*(act), *timeS*(s),tcl) ∧
Satisfy-Cons(*timeA*(act), *timeS*(s), tc2).

The other non-recursive rules handle basic temporal constraints. These rules represent a simple translation of each of the temporal constraint functions into the equivalent relation. The following axioms are all examples of the It constraint.

∀act, s.
Satisfy-Cons(*timeA*(act), *timeS*(s), *lt*(*disp*(B, t_1), *disp*(B, t_2))) if
begin(*timeA*(act)) + t_1 < *begin*(*timeS*(s)) + t_2

∀act, s.
Satisfy-Cons(*timeA*(act), *timeS*(s), *lt*(B, E)) if
begin(*timeA*(act) < *end*(*timeS*(s))

∀act, s,
Satisfy-Cons(*timeA*(act), *timeS*(s), *lt*(E, B)) if
end(*timeA*(ati)) < *begin*(*timeS*(s))

∀act, s.
Satisfy-Cons(*timeA*(act), *timeS*(s), *lt*(E, E)) if
end(*timeA*(act)) < *end*(*timeS*(s))

3.5 NORM VIOLATIONS

In order to make inferences about norm violation one need to know the existence of three fundamental things. The first thing is the existence of a normative token such as an agent's obligation or prohibition to carry out or desist from carrying out an action. The second thing is the failure to conform to the requirement of the normative token. The third thing is the validity of the normative rule at the time the condition held.

The violation of an obligation takes place when in the occurrence of the situation within the validity period of the rule that generated the normative token, the agent implicated is unable to carry out the needed action within the required time constraint. This is formalised as the following Horn's Clause:

∀a, norm-id, s.
Violate(a, norm-id, s) if
∃ act-type, tc.
Obligation(a, act-type, s, tc, norm-id) ∧
Validity(norm-id, j) ∧ Subinterval (*timeS*(s), j) ∧
∀act (¬EventType (act, act-type)∨
¬Actor(a, act) ∨
¬Satisfy-Cons(*timeA*(act), *timeS*(s), tc))

In the case of a prohibition, a violation takes place when in the case of the occurrence of the situation, the implicated agent carries out the forbidden action during a time interval that satisfies the constraint with the time of the situation. Such a violation is formalised thus:

∀a, norm-id, s.
Violate(a, norm-id, s) if
∃ act-type, tc, act.
Prohibition(a, act-type, s, tc, norm-id) ∧
ActionType(act, act-type) ∧
Actor(a, act) ∧
Validity(norm-id, j) a, Subinterval(*timeS*(s), j) ∧
Satisfy-Cons(*timeA*(a), *timeS*(s), tc).

This approach contrasts with timed violations that were implemented by Stratulat *et al*., (200la). In Stratulat *et al.* work, a violation of an obligation is said to have taken place at a time t, if an agent has an obligation to carry out an action during the time interval (t_1, t_2) and as at time t which is later than t_2 the action has not yet been taken.

3.6 COMPARATIVE EVALUTION OF CKRS WITH EXISTING NORMS REPRESENTATIONS

The proposed CKRS is concerned with the logic constructs of norms representation leading to normative violation inferences that will facilitate how multiagent sytems regulate the agents in the systems. Table I shows the comparison of the proposed CKRS with existing norms representation approaches in literature. The comparison shown in Table I involves norms representation literature between years 2001 to 2012.

ISSN 2229-5518

Table I Comparative Evalution of CKRS with Existing Norms Representations

AUTHOR (S)	NORMATIVE POSITION REPRESENTATION			META NORMS	TEMPORAL CONSTRAINTS					VIOLATION				
	LP and variant	EC	FOL		Quantitative	Qualitative	Fixed Time	Relative Time	Action	Situation Event	Situation Process	Situation State	Norm Identity	Norm Validity
Sadri *et al.*, (2006)	ALP ✓				N/A	N/A	✓							
Artikis *et al.*, (2009)		✓							✓					
Artikis and Sergot (2010)		✓							✓					✓
Stratulat *et al.*, (2001a)			✓		N/A	N/A	✓		✓					✓
Panagiotidi, *et al.*, (2009)	ASP ✓				✓	✓		✓						✓
CKRS			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

Key:
LP: Logic Programming
ALP: Abductive Logic Programming
ASP: Answer Set Programming
N/A: Not Applicable
CKRS: Condition Based Knowledge Representation Structure

The CKRS can represent real life norms much better than existing methods in literature because it was formalised using reified Frst Order Logic that confer expressive advantages in representing norms and meta-norms. Furthermore, the CKRS based its temporal constraints reasoning on Time Point Images (TPI) that is able to capture both qualitative and quantitative constraints between arbitrary times. However, CKRS allows the representation of norms using relative times. The violation inference of the proposed CKRS takes into consideration, the situation warranting the norm violation and the validity of the norm as the time of the situation and the time of the action with respect to whether it satisfies the prescribed constraint with the time of the situation. When a violation is established, the CKRS will explicitly capture the condition (situation) warranting the violation and the identity of the norm violated.

4 CONCLUSION

The proposed Temporal Constraints Structures' (TCS) can represent constraints that combine both qualitative and quantitative relationships. Also, the time point images (TPIs) involved in the TCS will aid greatly in relating the time of the condition of a norm (situation) with that of the effect of the norm which is the time of the action of the type specified. This is opposed to existing methods in literature which is based on the time of the action only. Furthermore, a condition-based knowledge representational structure was developed. The condition and the identity of the norm violated were adequately represented with the developed structure. Multi-agent systems can adopt this CKRS to regulate agents' interaction in the system. This kind of normative system will enhance the ability of agents new to the society to become aware of existing norms in the society.

5 RECOMMENDATIONS

A future direction of this work is to situate the violation necessarily within the context of a wider logical theory of action. Such a logical theory of action which is beyond the scope of this work will be mindful of the fact that an action has both *preconditions* and *post-conditions*. In that case for example, an agent may not be held liable for a violation if the conditions are not suitable for carrying out the obligation. In this type of scenario should be regarded as an *excusable violation*, which should not attract any sanction. Another potential direction of this work is to explore how plan recognition theories can enchance norm learning. Plan recognition theories enable the inference of high level goals from individual actions carried out by an individual agent.

REFERENCES

1. Ahmad, A., Ahmed, M., Yusof, M. Z. M., Ahmad, M. S. and Mustapha, A. 2011. Resolving conflicts between personal and normative goals in normative agent systems. In *7th International Conference on Information Technology in Asia: Emerging Convergences and Singularity of Forms (CITA'll)*, Kuching, Sarawak. 1-12.
2. Allen, J. F., 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* . 26.11:832-843.
3. Allen, J. F., 1984. Towards a General Theory of Action and Time. *Artificial Intelligence.* 23:123-54.
4. Allen, L." E., 1980. Language, law and logic: Plain drafting for the electronic age. In B. Niblett, editor, *Computer Science and Law*. 75-100.
5. Allen, L. E., 1982. Towards a normalized language to clarify the structure of legal discourse. In Deontic Logic, *Computational Linguistics and Legal Information Systems*. 2: 349-407.
6. Artikis, A. 2003. Executable specification of open norm-governed computational systems. Ph. D

Thesis. Department of Electrical and Electronic Engineering. University Of London. 1-235.

7. Artikis, A. 2009. Dynamic protocols for open agent systems. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) ACM* . 97-104.
8. Artikis, A., Sergot, M. and Pitt, J. 2007. An executable specification of a formal argumentation protocol. *Artificial Intelligent*. 171(10-15):776-804.
9. Artikis, A., Sergot, M. and Pitt, J. 2009. Specifying norm-governed computational societies. *ACM Transactionson Computational Logic*. 10(1): 2 - 34 .
10. Artikis, A. and Sergot, M. 2010. Executable Specification of Open Multi-Agent Systems, *mjs-IGPL-artser-fmal.tex*. 1-32.
11. Galton, A. P. 2005. Eventualities. In Fisher M, Gabbay D' and Vila L.(eds) *Handbook of Temporal Reasoning in Artificial Intelligence*, Elsevier. 1-23.
12. Operators vs arguments: the ins and outs of reification. *Kluwer Academic Publishers, Netherlands*. 1-23.
13. Hoffmann, M. J. 2003. Entrepreneurs and norm dynamics: an agent based model of the norm life cycle. *Technical Report, Department of Political Science and International Relations, University of Delaware, Newark, Del, USA*. 1-38.
14. Jones, A. J, I. and Kimborough S.O, 2012. On the Representation of Normative Sentences in FOL. In Artikis A. *et al.* (eds) *Logic, Programs, Norms and Actions: Essays in honour of Marek J. Sergot On the Occasion of his 60th* Birthday Lecture Notes In Artificial Intelligence, Springer Verlag, Berlin, 7360: 273-294.
15. Jones, A. J. I. and Sergot, M. 1993. On the characterisation of law and computer systems: The normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*, John Wiley and Sons. 275-307.
16. Kowalski R. and Sergot M., 1986. A logic-based calculus of events. *New Generation Computing*, 4.1:67-96.
17. L'opez y L'opez, F. 2003. Social power and norms: Impact on agent behavior. Ph.D thesis. Electronics and Computer Science Department, Faculty of Engineering and Applied Science, University of Southampto. xiii + 241.
18. Marin R. and Sartor 0. 1999. Time and norms: a formalisation in the event calculus. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*, ACM Press. 90 -100.
19. Panagiotidi S, Nieves, J.C, and Vazquez-Salceda, J. 2009. A framework to model norm dynamics in Answer Set Programming. *Proceedings of the Second Multi-Agent Logics, Languages, and Organisations Federated Workshops*, Turin, Italy. 1-23.
20. Royakkers, L. and Dignum, F. 1997. Giving permission implies giving choice. In *E. Schweighofer, editor, 8th International Conference and Workshop on Database and Expert Systems Applications, Toulouse, France*. 198-203.
21. Sadri, F., Stathis, K. and Toni, F. 2006. Normative KGP agents. *Computational and Mathematical Organization Theory*. 12.2-3:101-126.
22. Schubert, K.L. 2000. The situations we talk about. *From J. Minker (ed.), Logic-Based Artificial Intelligence, KluwerAcad. Publ, Dortrecht*. 407 - 439.
23. Stratulat, T. 1999 . Normative agent systems. In *Proceedings of Policy Workshop*. 1-5.
24. Stratulat, T. Cerin-Debart, F. and Enjalbert, P. (200la). Norms and time in agent-based systems. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*, ACM Press. 178-185.

25. Stratulat, T. Cerin-Debart, F. and Enjalbert, P. 2001b. Temporal reasoning: an application to normative systems. In *Proceedings of Symposium on Temporal Representation and Reasoning (TIME)*, IEEE Computer Society. 41-47.