Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 717–726

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Non-Functional Norms Specification and Verification Approach for Normative Multi-agents Systems

Ezzine Missaoui[a,*], Belhassen Mazigh[b], Vincent Hilaire[c], Sami Bhiri[d]

[a]*ENSI, University of Manouba, 2010, Manouba, Tunisia*
[b]*Department of Computer Sciences, FSM, Monastir, Tunisia*
[c]*CIAD UMR 7533, Univ. Bourgogne Franche-Comt, UTBM, F-90010 Belfort, France*
[d]*OASIS-ENIT, Tunis, Tunisia*

Abstract

Normative Multi-agent system (NMAS) forms a promising approach to software engineering for the development of autonomous systems, in norms-based environments. During the construction of these systems, different requirements can be considered: functional and non-functional requirement. Existing research in NMAS have mostly focused on the study of functional and behavioral requirements specification and verification, while Non-Functional Requirements (NFR) are crucial in the development of systems. They have dealt with the NFR implicitly through the capabilities of functional requirements. An important research issue refers to NFR in NMAS, i.e. how to specify and enforce NFR in NMAS that involves heterogeneous and autonomous agents. The NFR define constraints for the system and specify the qualities that a system can have, for example, security, performance, reliability, comfort, availability. Also, NFR might conflict among each other (for example, cost and quality, comfort and economy). A conflict between two NFR occurs when the fulfillment of one NFR violates another NFR. One of the main challenges currently faced in NMAS research is that of NFR specification and verification. In particular, how to specify and verify NFR in autonomous systems?

In this paper, we propose a normative approach that allows the specification and verification of NFR for autonomous systems. To do this, we propose a new type of norms, called Non-Functional Norms (NFN). Indeed, our approach provides (i) a BNF normative language for specifying NFN, and (ii) a mechanism, based on Constraint Satisfaction Problem (CSP) technique, for the detection and resolution of normative conflicts. Our normative approach is also illustrated by a case study describing Smart City Management System (SCMS).

© 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of KES International.

Keywords: Normative Multi-agents Systems; Autonomous System; Non Functional Requirements; Norms; Normative Approaches; Conflict Verification; Constraint Satisfaction Problem (CSP).

* Corresponding author. Tel.: +216 20 927 340
E-mail address: ezzine.missaoui@gmail.com

1877-0509 © 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of KES International.
10.1016/j.procs.2019.09.227

1. Introduction

Technological and scientific progress, particularly in the field of artificial intelligence, allows endowing the systems with some autonomy. Autonomy [1] is the ability of a system to perceive, analyze, communicate, make decisions, and act in order to achieve their objective. Normative multi-agent systems (NMAS) [2, 3] forms a promising approach to software engineering for the development of autonomous systems like intelligent transportation system [4] and smart city management system. In effect, NMAS consist of autonomous and heterogeneous agents regulated by norms [3], that interact to solve a problem or realize collectively a task. Norms are able to regulate agent behavior while allowing the agent to remain autonomous. So, one of the main challenges of NMAS is to both maintain the autonomy and heterogeneity of agents, and provide guarantees about the behavior and outcomes of individual agents and of the system as a whole.

During the construction of autonomous systems using NMAS, different requirements can be considered: functional and non functional requirement. Non-Functional Requirements (NFR) [5] describe the functioning of the system, while the Functional Requirements (FR) describe what the system should do. FR specify the behavior or function of the system when certain conditions are met. NFR are expressed as constraints and desirable properties for system behavior. They are also considered as quality attributes of a system. Existing normative approaches in NMAS have mostly focused on the study of functional and behavioral requirements specification and implementation, such as [6, 7, 8, 9, 10], while NFR are crucial in the development of autonomous systems. They have dealt with the NFR implicitly through the capabilities of the functional requirements, such as [8, 9]. For example, the authentification is a functional requirement, while the security is a NFR, where the authentification is a security capability. This means that the system must be secured and that it executes functions of security.

An important research issue refers to NFR specification in NMAS, i.e. how to express, specify and enforce NFR (constraints and limits) in NMAS based on autonomous agents, while maintaining the autonomy and heterogeneity of agents. One of the main solutions is to propose a new type of norms, called Non-Functional Norms (NFN), which can be considered as a powerful way of specifying the NFRs of autonomous systems. These norms are a compromise between the autonomy of the system entities and the NFRs applied to the functioning of these entities.

NFR can conflict with each other [11] (for example, cost and quality, comfort and economy). For example, to address security concerns, when you enter a smart building, you have several choices, among them, setting a two-level access code or to use biometrics. Using a two-level access code would conflict with usability concerns while the use of biometric devices might conflict with cost concerns. The second challenge in this paper is the detection and resolution of normative conflicts of NFR. However, we will propose a mechanism for verifying the conflict among NFR.

In this paper, we propose a normative approach that allows the specification and verification of NFRs of autonomous systems. This approach provides: (i) a Backus-Naur Form (BNF) normative language whose aim is to describe the NFNs that specify NFRs : we extend the BNF normative languages proposed in [12, 13] to represent NFRs, and (ii) a mechanism, based on Constraint Satisfaction Problem (CSP) technique and relationships between NFRs, to verify normative conflict between NFNs. Our approach is also illustrated by a case study describing the smart city management system (SCMS).

The rest of this paper is organized as follows: Section 2 presents a motivating example; In section 3, an overview of normatives approaches is presented; Section 4 details our NFN specification and verification approach. The last section concludes our paper and gives some future work.

2. Motivating Example

Let's consider a Smart City Management System (SCMS) as a motivating example which will be used throughout this paper to illustrate our normative approach for NFN specification and verification. Smart City represents the concept of efficient and sustainable city, capable of responding in the most adequate way to the basic needs of the city. This response to the needs is mainly based on the economic, environmental, social and operative fields. To do so, an efficient and lasting infrastructure must include: intelligent energy, smart buildings, intelligent transportation, smart security and safety, intelligent health care, tele-communications, and emergency services. It integrates the best existing concepts (materials, systems and technologies) to meet the requirements of managers and users. Smart City

has a lot of features: (i) improving comfort in buildings (heating, air conditioning, ventilation and electric lighting); (ii) providing surveillance and security of buildings; (iii) helping to reduce energy consumption.

Recently, the design theories of MAS allow solving problems in a distributed manner by taking advantage of social behaviors as well as the individual behavior of the agents and by sharing features, norms and objectives. Since a smart city requires the features of a autonomous system (social control, autonomy and heterogeneous), MAS have been applied in this environment with promising results, for example to address multiple aspects in the management of smart cities [14].

Smart city is composed of several heterogeneous and autonomous entities interacting with each other. In order to meet the needs of all these entities, smart city must anchor its development in respect of a set of NFR and shared agreements. SCMS is an autonomous system who requires social control. Developments of autonomous software for SCMS provide new challenges for requirements specification and verification. We define a set of norms that specify the NFR of SCMS. An informal specification of some norms for such systems is reported below:

Norm 1: Smart City is obliged to provide security of users.
Norm 2: Smart City is obliged to reduce energy consumption.
Norm 3: Smart City is obliged to improve comfort of users.
Norm 4: Smart City is obliged to fulfill services as soon as possible.

3. Related Work

3.1. Norms specification approaches

Several works on normative models, for multi-agent systems, focus on the specification aspects of norms, such as [12, 13, 15, 16, 17, 18].

In [12], Garcia-Camino et al. have proposed a BNF normative language to specify dialogical action that agents can hold to attain their goals. They have defined a normative language to specify obligations, permissions, prohibitions, violations and sanctions to restrict agents' dialogical actions. This normative language considers dialogical actions as the only kind of action performed by agents. Thus, it cannot be used to describe norms to regulate actions not (directly) related to the interaction between agents (non-dialogical actions). Viviane Torres da Silva [13] has extended the normative language proposed in [12] with the notion of non-dialogical actions proposed by Vazquez-Salceda et al. [15]. Other, normative model was proposed in [16], to introduce two new types of norms: internal (private) and external (public) norms. In [17], authors proposed a normative model called Moise+. This last is based on the concepts of missions (a set of global goals) and global plans (the goals in a structure). A mission is a set of coherent goals that an agent can commit to. The Moise+ model specifies a set of constraints for agents along three dimensions: a structural, functional and deontic specification. Recently, new proposal on normative models was made to design methodologies for the analysis and design of normative open MAS, like the ROMAS methodology [18]. This methodology is focused on the analysis and design processes for developing open organizational MAS where agents interact by means of services, and where social and contractual relationships are formalized using norms.

These works have mostly focused on the study of functional and behavioral requirements specification and implementation, while NFR are crucial in the development of autonomous systems. They have dealt with the NFR implicitly through the capabilities of functional requirements. Our normative approach proposes a BNF normative language whose aim is to explicitly describe the non functional norms (NFNs) that specify NFRs.

3.2. Conflict verification approaches

NFNs that specify NFRs can conflict each other. A conflict between norms is a situation in which the fulfillment of a norm causes a violation of another one. According to normative conflict verification, several techniques and tools have been proposed for detection and resolution of conflicts between norms. In what follows, we present some of the most relevant search results like: ROMAS CASE tool [19], OWL-POLAR [20, 21], Unification and constraints [22].

In [19], Garcia et al. present the ROMAS CASE tool, which is based on model-checking techniques to check the coherence of a modeled legal context. The ROMAS CASE tool can detect direct conflicts among organizational and agent norms. In this tool, the verification module has been developed using the SPIN model checker [23].

The work described in [20] proposes a POLAR Agent application that is able to detect conflicts. The conflict detection mechanism uses substitution and takes into account relationships of entity specialization. The authors provide an automated support for identifying and resolving logical and functional conflicts. In [21], Sensoy et al. provide a mechanism to treat conflicts among policies with OWL-POLAR (Ontology Web Language-based POlicy Language for Agent Reasoning) [24]. This mechanism provides means to determine the context in which different policies may conflict. The authors propose an OWL-based representation of policies that use a novel combination of ontology consistency checking and query answering. They propose a detection algorithm to identify conflicting norms by anticipating contexts in which conflicts may arise.

Vasconcelos et al. [22] propose a mechanism to detect normative conflicts between prohibitions and obligations or permissions. They use first-order unification to check if the variables of a prohibition overlap with the variables of an obligation or permission. The authors use unification and constraint satisfaction to detect potential conflicts between norms. To resolve normative conflicts, Vasconcelos et al. propose norms reduction algorithms for adoption and elimination of norms to maintain freedom of conflict.

All these approaches are associated with activation and deactivation conditions, that can be a state, such as a date (start date and end date). A conflict occurs when the variables of norms have overlapping values. Existing approaches allow the detection of the activation period of the norms overlap based on the unification technique and allow the resolution of conflict removing any overlap in their values. Whereas in our case norms are associated with NFRs, so they are still applicable. Therefore, these proposed techniques do not allow the detection and resolution of normative conflict related to NFRs. We propose the use of the CSP [25] technique to solve this problem.

4. Non-Functional Norms specification and verification approach

During the construction of autonomous systems, different requirements can be considered: Non-Functional and Functional requirements. At the society level, the system is seen as a whole that wants to meet functional needs by respecting non-functional requirements. In this section, we propose a new normative approach for specifying and verifying non-functional requirements of autonomous systems. Our approach provides both a formalism for specifying NFRs using a set of NFNs, and a mechanism for verifying conflict among these norms. First, we will clarify how we specify the NFRs of autonomous systems using NFNs. Then, we consider the conflict verification among these NFNs using constraint satisfaction problem (CSP) technique.

4.1. Specifying non-functional requirements using non-functional norms

Initially, we propose a BNF normative language whose aim is to describe the non-functional norms that specify the NFR of the system. We extend the BNF normative languages proposed in [26, 27] to represent NFR [5], to describe the system behavior and the constraints of its features and to specify the characteristics or quality attributes of the system. In [26], Garcia-Camino et al. have proposed a BNF normative language to specify dialogical action that agents can hold to attain their goals. They have defined a normative language to specify obligations, permissions, prohibitions, violations and sanctions to restrict agents' dialogical actions. This normative language considers dialogical actions as the only kind of action performed by agents. Thus, it cannot be used to describe norms to regulate actions not (directly) related to the interaction between agents (non-dialogical actions). Viviane Torres da Silva [27] has extended the normative language proposed in [26] with the notion of non-dialogical actions proposed by Vazquez-Salceda et al. [28]. These works have mostly focused on the study of functional and behavioral requirements specification of agents. We extend these BNF normative languages to support NFNs.

The normative languages which we use for the norms description is based on the deontic logic [29, 30]. We start with a describing norms related to non-functional requirements, then we describe the grammar. The norms that can be specified by using the language are regulative norms [31] that state obligations, permissions or prohibitions.

4.1.1. Non-Functional Requirements

Requirements are frequently divided into Functional requirements, what the system should do, and non functional requirements, how the system should be [32]. Non functional requirements specify quality attributes of the system and/or constraints and are not easdily expresses as transformations/functions. There are several types of NFR. The

one we are interested in this paper is often named Execution qualities, as for example safety, security and usability, which are observable during operation (at run time).

NFR describe the behavior of a system and the limits of its functionality. They are expressed as constraints (restrictions on one or more values) and desirable properties (qualities) of the system such as its performance (energy saving, response time, throughput), availability, reliability, interoperability, security, regulatory, comfort, etc.

These properties can be classified in two categories, namely qualitative properties (security, comfort, etc.) whose values are symbolic and quantitative properties (response time, availability, energy saving, etc.) whose values are numerical. It is necessary to quantify these last properties with metrics that can be measured and tested. The transformation of a qualitative property to a quantitative property consists of associating its characteristics with numerical values. For example, the qualitative property ”comfort of smart buildings” can be defined by the following five modalities: excellent, very good, good, medium, bad and very bad. The digital encoding of the property can be done as follows: ”6 = excellent”, ”5 = very good”, ”4 = good”, ”3 = medium”, ”2 = bad” and ”1 = very bad”. Similarly, the security property can be defined by a set of modalities that describe the different rates of security: excellent state, good state, normal state, and emergency state. The digital encoding of the property can be done as follows:”1 = excellent state”, ”2 = good state”, ”3 = normal state”, and ”4 = emergency state”. The same approach will be used for others qualitative properties.

In the context of this paper the proposition is to use norms to specifiy NFR. Indeed, as illustrated in the example these kind can be expressed in terms of predicates, equalities or inequalities for example

4.1.2. Specifying Non-Functional Norms

In some description of normative language, norms are defined as the composition of a deontic concept (characterizing obligation, prohibition or permission) and an action followed by a temporal situation which indicates the period during which norm is activated.

In our proposal, norms define the system's properties (security, performance, comfort, quality of service, etc.). They are expressed as the composition of a deontic concept (characterizing obligation, prohibition or permission) and a property (the object of the norm). It assigns a deontic value to a property. The properties to which norms can affect a deontic value are modeled by parameterized predicates.

Formally a non functional norm related to NFR is defined as follows: **DC(Prop(Var, Val))**, with:

- **DC =** $\{O, I, P\}$: deontic concept, where O: Obligation, I: Prohibition and P: Permission.
- **Prop**: represents a property (quality attribute),
- **Prop(Var, Val)**, Predicate, that deals with a couple of objects to represent an equivalence or order relation,
- **Var**: denotes an entity (role, agent, organization) on which is defined the predicate Pred.
- **Val**: represents the required value for the entity Var.

For example, in our case study, SCMS, we can associate a deontic concept on the energy saving property with the following norm:
Norm 1, as defined in section 2: Smart City is obligated to reduce energy consumption (the energy consumption per day must be lower than a given value, for example 100 KW per day).
And it is formally written in the form **O(energy_saving(Smart_City, 100))**.

Two types of predicates can be defined based on a property: an Upper Bound Predicate (UBP) defines according to a property that expresses a maximum limit as the highest level, and a Lower Bound Predicate (LBP) defines according to a property that expresses a minimum limit as the lowest level. Therefore, UBP concern the properties for which we try to minimize (cost, price, energy consumption, etc.), and LBP concern the properties for which we try to maximize (security, comfort, availability, etc.).

Compared to our example of motivation, the predicates Pred1, energy_economy (smart_city, 500), and Pred2, security (smart_city, 2), are predicates of UBP types and the values 500 and 2 are respectively considered the highest level of energy consumption and security in smart city. While predicates Pred3, comfort (smart_city, 4) and Pred4, availability (smart_city_services, 0.9) are predicates of LBP types and the values 4 and 0.9 are respectively considered as the lowest level of comfort and availability of services in smart city.

4.1.3. Grammar

The specification of non functional norms related to non functional requirements is given by the following grammar:

$< norm >::=< deontic\_concept > (< formula >)$
$< deontic\_concept >::= O|I|P$
$< formula >::=< atom >$
$\quad | < formula >< connector >< formula >$
$\quad | < quantifier >< variable >< formula >$
$< atom >::=< predicate > (< term >, < term >, ...)$
$< term >::=< constant > | < variable >$
$\quad | < function > (< term >, ...)$
$< connector >::= \neg | \Rightarrow | \Longleftrightarrow | \bigwedge | \bigvee$
$< quantifier >::= \exists | \forall$

Formula is a predicate that specifies a property. Terms can be variables (represent the entities involved in the specified system, i.e., agents or organizations in MAS), or constants (values of desirable properties, i.e., preferred comfort level in an organization equal to 5). The predicate is a property related to a variable (entity), to express its relation to a given value.

We show how to use this grammar through several examples related to our case study SCMS. This system has several non-functional requirements (system quality), that define the system's properties like: security, comfort, economy of energy, performance, quality of service, availability, response time, etc. By using the above grammar, it is possible to describe NFN that specify NFRs of SCMS:

Norm 1: O(security(smart_city, 2))
Norm 1 means that smart city is obliged to improve security of users, since security is a qualitative property, we have transformed it into a quantitative property. Also security is defined by an upper bound predicate, so this norms means that it is obliged that the security in smart city does not exceed 2, less than or equal to 2.
Norm 2: O(energy_saving(smart_city, 500)) /*Smart city is obliged to not exceed 500 kw per day.*/
Norm 3: O(comfort(smart_city, 3)) /*Smart city is obliged to improve comfort of users.*/
Norm 4: O(response_time(Smart_city, 10)) /*Smart city is obliged to satisfy a service in a delay not exceeding 10 seconds.*/

4.2. Conflict verification

Non-functional requirements (NFR) in software engineering presents a systematic and pragmatic approach to building quality into software systems. NFRs can conflict each other (for example, cost and quality, comfort and economy). Conflicts between NFRs mean that achieving one requirement can impact another one. These conflicts happen when a procedure favors the first characteristic but create difficulties for the second. For example, to ensure security concerns, when entering a smart building, one might have several choices, among them, setting two level access code or to use biometrics. Using a two level of an access code would conflict with usability concerns while the use of biometric devices might conflict with cost concerns. Another example, satisficing security might call for the use of some encrypting mechanism, but the use of this mechanism might conflict with performance needs. To achieve absolute satisfaction among NFRs it is important to detect and analyze conflicts between them. We start by identifying the relationships between NFRs, then we use CSP formalism to detect normative conflicts among NFRs.

4.2.1. Identifying relationships between NFRs

Several models have been published in literature to represent positive or negative relationships among NFRs, [33, 34]. Relationships focus on how NFRs contribute negatively to the other NFRs. For instance, Wiegers and et al. [33] present a matrix of positive and negative relationships among NFRs, and Sadana and Liu [34] also present a model of relationship among quality attributes. These models illustrate some typical interrelationships among NFRs.

Figure 2 illustrates a matrix of positive and negative relationships among pairs of NFRs [33]. A minus symbol means that influencing the NFR in a row can affect the NFR presented in the column. For example, Security impacts

on Performance. Based on these relationships between NFRs, we can detect and resolve normative conflicts using the CSP technique.

A key challenge in dealing with NFRs is that there is no obvious means of deciding when they are satisfied. In effect, these NFRs are never fully satisfied, but satisfied to varying degrees. We use CSP technique to deal with dependencies among NFRs since they frequently involve many conflicts among possible solutions to satisfice one or more NFR. CSP allows to trade-off varying degrees of satisfaction of contradictory non-functional requirements. It identifies and analyzes conflicts between NFRs based on relationships among quality attributes.

	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability								+		+						
Efficiency	+				−	−	+	−			−		+		−	
Installability	+								+					+		
Integrity			−		−		−			−		+		+	−	−
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−			+				+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

Fig. 1. Relationships between NFRs [33].

4.2.2. Modelling Non Functional Norms as a CSP :

A CSP is a formalism to represent a problem in the form of a set of variables and constraints. Variables take their values in a domain that can be discrete or continuous. Constraints can be unary, binary, or n-ary. In order to solve a CSP, the goal is to assign a value to each variable so that all constraints are satisfied. Formally a CSP is a triplet (X, D, C) where:

- X = $\{X_1, X_2, \ldots, X_N\}$ is a finite set of N variables,
- D = $\{D_{X_1}, D_{X_2}, \ldots, D_{X_N}\}$ is the function that associates to each variable X_i its domain D(Xi), i.e. the set of values that X_i can take,
- C = $\{C_1, C_2, \ldots, C_m\}$ is a finite set of m constraints. Each constraint C_i is a relationship between certain variables of X, restricting the values that these variables can take simultaneously.

To represent NFN that specify NFR with the CSP formalism, we propose the following correspondences:

- Each variable X_i represents a property P_i (quality attribute) in N, for example (performance, comfort, security, reliability, availability, ...).
- Each property Pr_i has a value domain D_{Pr_1}, associated with each property, for example the security property is defined by the following four modalities that describe the different rates (level) of security, $\{1 =' excellentstate', 2 =' goodstate', 3 =' normalstate', and 4 =' emergencystate'\}$.
- Each constraint C_i relates to a subset of variables in P, corresponds to the conjunction of norms, DC(Pr (Var, Val)), associated with the properties Pr that are related.

Our case study, SCMS, has several NFRs (properties), such as: security, response time, performance, comfort, economy of energy, cost, availability, etc. We define NFN that specify NFRs of SCMS with CSP formalism as follow:

- X_1: Security, $D_{X_1} = \{SL_1, SL_2, ..., SL_5\}$, with SL: Security Level.
- X_2: $Response\_Time$, $D_{X_2} = \{t_1, t_2, ..., t_5\}$, with t: time.

- X_3: Performance, $D_{X_3} = \{PL_1, PL_2, ..., PL_5\}$, with PL: Performance Level.
- X_4: Comfort, $D_{X_4} = \{CL_1, CL_2, ..., CL_5\}$, with CL: Comfort Level.
- X_5: *energy_saving*, $D_{X_5} = \{CI_1, CI_2, ..., CI_5\}$, with CI: Consumption Interval.
- X_6: Cost, $D_{X_6} = \{C_1, C_2, ..., C_5\}$, with C: Cost.
- X_7: availability, $D_{X_7} = \{a_1, a_2, ..., a_5\}$, with a: availability.

Now, we use the relationships matrix [33] to define constraints C_i on properties that are related.

- $C_1 = (X_1 = SL_1 \Rightarrow X_4 = t_1) \vee ... \vee (X_1 = SL_5 \Rightarrow X_4 = t_5)$ /*Security Level depends on the property of *Response_Time**/
- $C_2 = (X_4 = t_1 \Rightarrow X_7 = PL_1) \vee ... \vee (X_4 = t_5 \Rightarrow X_7 = PL_5)$ /*The property of *Response_Time* depends on the property of performance*/
- $C_3 = (X_7 = PL_1 \Rightarrow X_3 = CL_1) \vee ... \vee (X_7 = PL_5 \Rightarrow X_3 = CL_5)$ /*The property of performance depends on the property of Comfort*/
- $C_4 = (X_3 = CL_1 \Rightarrow X_2 = CI_1) \vee ... \vee (X_3 = CL_5 \Rightarrow X_2 = CI_5)$ /*The property of comfort depends on the property of *energy_saving**/
- $C_5 = (X_2 = CI_1 \Rightarrow X_5 = C_1) \vee ... \vee (X_2 = CI_5 \Rightarrow X_5 = C_5)$ /*The property of *energy_saving* depends on the property of Cost*/

4.2.3. Conflict detection

After the representation of non functional norms that specify NFRs with the CSP formalism, we use the CSP algorithms to check for the existence or not of conflicts between the system's properties. CSP allows us to detect if the norms are in normative conflict and to detect all the properties that are under the influence of these norms. Conflict detection can be done at design time to avoid conflicts during execution, or at runtime wherein agents must be able to resolve conflicts dynamically.

Solving a CSP requires finding for each variable X_i a value in D(X_i) that is consistent with C (i.e. which does not violate any constraints of C). A solution S of a CSP (X, D, C, R) is a coherent instantiation of all the variables of the problem. A CSP is said consisting if and only if it has at least one solution. If no solution exists, then the CSP is unsatisfiable, inconsistent.

When we formalize our application problem as a CSP, we do not have to develop algorithms to solve it from zero, because various algorithms for the resolution of CSP have been developed. We use existing filtering algorithms [35] to exploit the constraint definition in the CSP to remove values from domains that can not belong to any solution. Then, we use the chronological Backtracking (BT) algorithm as the basic algorithm for CSP resolution. In BT, variable X_i is chronologically assigned in a way that all the constraints are satisfied. Let us suppose that variable X_0 to X_i is assigned. The next variable that must be assigned will be the variable X_{i+1}. When it is impossible to assign a variable because all its values are inconsistent (an inconsistent value is a value leading to the violation of at least one constraint) with the variables previously assigned, the value of the last variable assigned will have to be modified. The algorithm BT ends in two different ways: either all the variables are assigned and the CSP is resolved (no conflict betwenn NFRs), or BT will have traveled the entire search tree and the CSP is inconsistent, and in this case we go back to the specification phase to correct the model.

4.2.4. Conflict Verification Algorithm based on CSP

We propose an algorithm, Algorithm 1, that allows to check the normative conflict related to NFN, based on CSP formalism. Our algorithm takes as input a normative context (set of norms) and give as output a Boolean variable that indicates if there is or not a conflict. To detect and resolve conflict, our algorithm reuses the existing algorithms of CSP. For the reason of the number of pages of the document (limited to 10 pages), we do not detail the chronological Backtracking (BT) algorithm which already exists in the literature. We use existing resolution algorithms to exploit the constraint definition in the CSP to remove values from domains that can not belong to any solution.

Algorithm 1 contains two parts. One part (line 13) makes it possible to check the direct conflicts between two pairs of norms. The second part of the algorithm (lines 16 and 17) uses the chronological Backtracking (BT) algorithm which already exists in the literature to check indirect normative conflicts between different non-functional norms. It

allows to perform a depth-first search in the search tree of CSP. At each node, BT tries to assign a value to a new variable. At each assignment, BT checks if the partial instantiation obtained satisfies all the constraints. if so, BT goes to the next variable. Otherwise BT tries another value if such a value exists. Otherwise BT goes back to the last assignment.

Algorithm 1: Conflict Verification Algorithm

Input: NC ={ N_i } : Normative Context (set of norms)
N, N' : Norm
N = DC(Prop(Var, Val)), N'= DC'(Prop'(Var', Val'))
DC, DC'∈ { O, I, P } : Deontic Concepts
Prop(Var, Val), Prop'(Var', Val') : Predicates
Prop: property, Var : term, Val : Constant Value
C = (N, N'): Couple of Norms
X={$Prop_1, ..., Prop_n$}, D={$D_{Prop_1}, ..., D_{Prop_n}$}
C = {$c_1, ..., c_n$}, I: Instantiation
Output: Conflict = { TRUE, FALSE } : BOOLEAN
Conflict = FALSE
Begin
  For ***all*** *C in NC* **do**
    If *Prop(Var, Val)==Prop'(Var', Val') and (DC!=DC')* **then**
      Conflict = TRUE
    end if
  end for
  If *(Conflict == FALSE)* **then**
    I = BT(X, D, C, I)
  end if
  If *(I == ∅)* **then**
    Conflict = TRUE
  end if
End

Algorithm 2: Function BT(X, D, C, I)

Begin
  If $X == \emptyset$ **then**
    return (I)
  end if
  Choose x ∈ X
  X = X \ {x}
  While D_x *!=* ∅ **do**
    Choose v ∈ D_x
    If *I ∪ (x, v) satisfies C* **then**
      I' = I ∪ (x, v)
      I' = BC(X, D, C, I')
    end if
    If *I' !=* ∅ **then**
      return (I')
    end if
    $D_x = D_x \backslash \{v\}$
  end while
  return (∅)
End

5. Conclusions and Future Work

In this paper, we have proposed a normative approach for non-functional requirements specification and verification. To do this, we have proposed a new type of norms, called Non-Functional Norms (NFN), that specify NFRs. This approach allows the NFN specification and verification of critical autonomous systems. Our approach provides both (i) a formalism for specifying non-functional requirements (system quality) using a set of NFNs, and (ii) a mechanism for the conflict detection and resolution of these NFNs, using CSP techniques.

Verification approaches are limited by the state-space of the system under study. Specifying and verifying a global normative model in a single level is then more complex and even NP-hard [36]. However, most normative approaches for multi-agent systems do not take into account the complexity of coherence verification algorithms of norms. As future work, we will propose a coherent refinement process of global norms, to reduce the complexity of the coherence checking of these norms. We will propose a set of generic refinement rules [37] that allow to decompose the global norms while respecting the coherence of the normative system.

References

[1] Huang, H.M., Pavek, K., Novak, B., Albus, J. and Messin, E. (2005). A framework for autonomy levels for unmanned systems (alfus). Proceedings of the AUVSIs Unmanned Systems North America, 849863.
[2] Hollander, C.D. and Wu, A.S. (2011). The current state of normative agent-based systems. Journal of Artificial Societies and Social Simulation 14, 6.

[3] Mahmoud, M.A., Ahmad, M.S., Mohd Yuso , M.Z. and Mustapha, A. (2014). A review of norms and normative multiagent systems. The Scientific World Journal.
[4] Tchappi, I. H., Galland, S., Kamla, V. C. and Kamgang, J. C. (2018). A brief review of holonic multi-agent models for traffic and transportation systems. Procedia computer science, 134, 137-144.
[5] Chung, L., Nixon, B.A., Yu, E. and Mylopoulos, J. (2012). Non-functional requirements in software engineering. volume 5. Springer Science and Business Media.
[6] Tinnemeier, N., Dastani, M. and Meyer, J.J. (2009). Roles and norms for programming agent organizations, in 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pp. 121128.
[7] Viana, M., Alencar, P. and Lucena, C., (2016). A modeling language for adaptive normative agents, in: Multi-Agent Systems and Agreement Technologies. Springer, pp. 4048.
[8] [15] Hbner, J.F., Sichman, J.S. and Boissier, O. (2002). Moise+: towards a structural, functional, and deontic model for mas organization, in: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, ACM. pp. 501502.
[9] Garcia, E., Giret, A. and Botti, V. (2015). Romas methodology, in: Regulated open multi-agent systems (ROMAS). Springer, pp. 5195.
[10] Missaoui, E., Mazigh, B., Bhiri, S. and Hilaire, V. (2017, June). Ncrio: A normative holonic metamodel for multi-agent systems. In International Conference on Hybrid Artificial Intelligence Systems, Springer, pp. 638-649.
[11] Mairiza, D., Zowghi, D. and Gervasi, V. (2013). Conflict characterization and analysis of non functional requirements: An experimental approach, in 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT), IEEE. pp. 8391.
[12] Garca-Camino, A., Noriega, P. and Rodrguez-Aguilar, J.A. (2005). Implementing norms in electronic institutions, in: 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 667673.
[13] da Silva, V.T. (2008). From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. Autonomous Agents and Multi-Agent Systems 17, 113155.
[14] Roscia, M., Longo, M. and Lazaroiu, G.C. (2013). Smart city by multi-agent systems, in: 2013 International Conference on Renewable Energy Research and Applications (ICRERA), IEEE. pp. 371376.
[15] Vzquez-Salceda, J., Aldewereld, H. and Dignum, F. (2004). Implementing norms in multiagent systems, in: Multiagent System Technologies, Second German Conference, MATES, pp. 313327.
[16] Missaoui, E., Mazigh, B., Bhiri, S. and Hilaire, V. (2017). A Normative Model for Holonic Multi-agent Systems. In International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, pp. 1251-1258.
[17] Hbner, J.F., Sichman, J.S. and Boissier, O. (2002). Moise+: towards a structural, functional, and deontic model for mas organization, in: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, ACM. pp. 501502.
[18] Garcia, E., Giret, A. and Botti, V., (2015). Romas methodology, in: Regulated open multi-agent systems (ROMAS). Springer, pp. 5195.
[19] Marques, M.E.G., Boggino, A.S.G. and Botti, V. (2013). A model-driven case tool for developing and verifying regulated open mas, in: Science of Computer Programming, Elsevier. pp. 695704.
[20] Aphale, M.S., Norman, T.J. and Sensoy, M. (2012). Goal-directed policy conflict detection and prioritisation, in: International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems, Springer. pp. 87104.
[21] Sensoy, M., Norman, T.J., Vasconcelos, W.W. and Sycara, K. (2012). Owl-polar: A framework for semantic policy representation and reasoning. Web Semantics: Science, Services and Agents on the World Wide Web 12, 148160.
[22] Vasconcelos, W.W., Kollingbaum, M.J. and Norman, T.J. (2009). Normative conflict resolution in multi-agent systems. Autonomous agents and multi-agent systems 19, 124152.
[23] Holzmann, G.J. (2004). The SPIN model checker: Primer and reference manual. volume 1003. Addison-Wesley Reading.
[24] Sensoy, M., Norman, T.J., Vasconcelos, W.W. and Sycara, K. (2010). Owl-polar: Semantic policies for agent reasoning, in: International Semantic Web Conference, Springer. pp. 679695.
[25] Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. Information sciences 7, 95132.
[26] Garca-Camino, A., Noriega, P. and Rodrguez-Aguilar, J.A., (2005). Implementing norms in electronic institutions, in: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM. pp. 667673.
[27] da Silva, V.T. (2008). From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. Autonomous Agents and Multi-Agent Systems 17, 113155.
[28] Vzquez-Salceda, J., Aldewereld, H. and Dignum, F. (2004). Implementing norms in multiagent systems, in: German Conference on Multiagent System Technologies, Springer. pp. 313327.
[29] Meyer, J.J.C. and Wieringa, R.J. (1993). Deontic logic in computer science normative system specification .
[30] Dastani, M., Meyer, J.J.C. and Grossi, D. (2013). A logic for normative multi-agent programs. Journal of Logic and Computation 23, 335354.
[31] Boella, G. and van der Torre, L.W. (2004). Regulative and constitutive norms in normative multiagent systems. KR 4, 255265.
[32] BKCASE Editorial Board. (2017). The Guide to the Systems Engineering Body of Knowledge (SEBoK).
[33] Wiegers, K. and Beatty, J. (2013). Software requirements. Pearson Education.
[34] Sadana, V. and Liu, X.F. (2007). Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements, in: 31st annual international computer software and applications conference (COMPSAC 2007), IEEE. pp. 215218.
[35] Dechter, R. and Cohen, D. (2003). Constraint processing. Morgan Kaufmann.
[36] Serramia, M., Lopez-Sanchez, M., Rodrguez-Aguilar, J.A., Rodrguez, M.,Wooldridge, M., Morales, J. and Ansotegui, C. (2018). Moral values in norm decision making, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018, pp. 12941302.
[37] Missaoui, E., Mazigh, B., Bhiri, S. and Hilaire, V. (2018). A Decomposition-based Approach of Global Norms for Hierarchical Normative Systems. Procedia Computer Science, 126, pp. 778-787.