Verifying Normative System Specifications Containing Collective Imperatives and Deadlines

(Extended Abstract)

Luca Gasparini
Timothy J. Norman
Martin J. Kollingbaum
Liang Chen
Dept. of Computing Science
University of Aberdeen, UK

John-Jules Ch. Meyer
Information and Computing Sciences
Utrecht University
The Netherlands.

ABSTRACT

Our focus is on the specification and verification of normative systems that include contrary-to-duty, collective and event-driven imperatives with deadlines. We propose an operational syntax and semantics for the specification of such systems. Using Maude and its model checker, we show how Linear Temporal Logic properties can be verified, and provide some experimental results.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model Checking

Keywords

Model Checking, Normative Systems, Collective Imperatives

1. INTRODUCTION

Existing approaches for the verification of normative systems consider limited representations of norms, often neglecting event-governed norms, collective imperatives, deadlines and contrary-to-duty (CTD) obligations. In order to capture the requirements of real-world scenarios, these structures are important. In this paper we propose methods for the specification and formal verification of such complex normative systems. We illustrate, through a intelligence, surveillance and reconnaissance (ISR) scenario, an operational syntax for CÒIR, a normative specification language that supports all the features of interest. We illustrate the operational semantics for a monitoring component that, given a set of CÒIR norms and a model of the environment and the agents acting within it, keeps track of activation, expiration, fulfilment and violations of norm instances. We use Maude [1] to implement this model and, use its Linear Temporal Logic (LTL) model checker to show how liveness and safety properties can be verified. We then provide some experimental results of model checking performance with varying numbers of agents in the ISR scenario.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

2. ISR SCENARIO

Consider a coalition of agents of the sea-guard, consisting of a set of *UAVs*, *helicopters*, and *boats*. The goal is to monitor and intercept unauthorized boats in a restricted area. The norms that guide the behaviour of the coalition are: **(1)** At any moment at least one member of the coalition must monitor the area. Moreover, we prefer UAVs to monitor the area than helicopters. We assume that only helicopters and UAVs are capable of monitoring. **(2)** Whenever an unauthorized boat is detected in the area, an agent must intercept it before the deadline (3 time steps) expires. **(3)** If no agent intercepts the boat, then at least one member of the coalition must send a report to head-quarters before the deadline (3 time steps) expires.

These are all examples of collective imperatives: they require *at least one member* of the coalition to act [3]. Norm 3 is also a CTD obligation that is activated in the event of a violation of obligation 2. Moreover, norms 2 and 3 require agents to perform an action *before a certain deadline* (a liveness property), norm 1 requires that *at any given moment* someone is monitoring the area (a safety property).

3. CÒIR

We now give a precis of the syntax and semantics of CÒIR. Compliance with norms is evaluated against a knowledge base KB that models the environment and the agents acting within it. A norm `ndi` is defined as a tuple:

$$\langle id_i, mod_i, act_i, exp_i, goal_i, ddl_i \rangle$$

where: id_i is an identifier; mod_i specifies the modality; obligation with deadline (O) or prohibition (F); act_i describes a pattern that, when matched in KB, causes a norm instance to be detached; $goal_i$ represents the situation that needs to be brought about (in the case of an obligation) or avoided (in the case of a prohibition); exp_i is a condition that, when met, causes the expiration of the norm instance; and the deadline for the fulfilment of the norm requirements (ddl_i) can be either temporal or symbolic and is defined only for obligations. A prohibition specifies that $goal_i$ must be false for the whole duration of the instance activation, whereas an obligation is fulfilled if the goal becomes true at a certain point before the deadline expires.

In order to enable the specification of CTD norms, we include the description of previous violations in KB. These

Table 1: Unbounded model checking results

cA	uB	nd1	nd2	nd3	nd4	States	Time
Part a: ddl2 = ddl3 = TEMPORAL (3)							
2	2	✓	✓		✓	20012	2m
3	2	✓	✓			19032	2m
2	2	✓	✓	✓		61884	10m
3	2	✓	✓		✓	72327	15m
2	2	✓	✓	✓	✓	243994	1h,16m
3	2	✓	✓	✓	✓	870165	25h
Part b: ddl2 = ddl3 = TEMPORAL (1)							
3	2	✓	✓	✓	✓	75245	16m

can then be referred to in the activation condition. We allow $goal_i$, exp_i, and ddl_i to include variables that are *not bound at activation time*. The combination of negation and existential quantification over these variables enables us to express some common patterns of collective norms; e.g. each member of a group g must perform a task t.

The semantics of the CÒIR monitoring component is given by means of structural operational semantics (SOS) [4]. A configuration contains the knowledge base KB, the set of norm descriptions, the set of active instances and the set of current violations. We organize the SOS in different stages: **(A)** Deactivate instances for which the expiration condition holds or the obligation has been fulfilled. **(B)** Check for violations of active obligations and prohibitions. **(C)** Check for the activation of new instances. It might be necessary to loop multiple times through A, B and C, for example to verify whether a newly activated instance is instantly violated.

Following Lamport [2], we model temporal deadlines by introducing a timer for each active obligation instance, avoiding to explicitly represent the current time which would lead to an infinite state space.

We chose Maude to implement the model because its rewriting rules syntax and semantics are very close to SOS, and in so doing we obtain an executable specification on which we can perform LTL model checking.

4. MODEL CHECKING CÒIR

Norm 2 from our ISR scenario can be specified in CÒIR as:

$nd2$ = ⟨ 2, O, $act2$, $exp2$, $goal2$, `TEMPORAL(3)` ⟩
$act2$ = `IN{ type(?add,coalition) /\ type(?ar,rArea) /\ inArea(?ag1,?ar) } FILTER NOT EXISTS{ type(?ag1,?type) /\ subType(?type,authAgent) }`
$exp2$ = `VIOLATED \/ NOT EXISTS { inArea(?ag1,?ar) }`
$goal2$ = `EXISTS{ intercepting(?ag2,?ag1) /\ memberOf(?ag2,?add) }`

Variables are identified by strings starting with a "`?`" character. The obligation is addressed to a coalition `?add`, and it is activated when an unauthorized agent `?ag1` is detected in a restricted area `?ar`. It is fulfilled if a member `?ag2` of `?add` intercepts `?ag1` before the deadline expires. An instance expires if it is violated or `?ag1` exits from `?ar`. Norm 1 can be specified as two norms: it is prohibited that no UAV is monitoring the area; and it is prohibited that neither a UAV nor a helicopter is monitoring the area.

The Maude model checker, given an initial state and a set of transition rules generates a Kripke structure that is used to verify a given LTL property. We now discuss how LTL model checking can be used to verify correctness-related properties. Consider a variation of `nd2` stating that we require one and only one agent to intercept an unauthorized boat. We may attempt to express the norm with the goal:

$goal2$ = `COUNT ( ?ag2 IN { memberOf(?ag2,?add) /\ intercepting(?ag2,?ag1) } ) = 1`

This goal is satisfied when exactly one member of the coalition (`?ag2`) is intercepting `?ag1`. To verify if this is correct we can use the following LTL property, which says that having both a UAV and an helicopter intercepting `ub` (an unauthorized boat) always results in a violation of `nd2`.

$$\Box((\texttt{intercepting(uav,ub)} \wedge \texttt{intercepting(heli,ub)} \wedge \texttt{inArea(ub,rArea)}) \rightarrow \texttt{violated(2)})$$

The model checker returns `false` and an example of an execution trace that violates the property. In fact, if `uav` and `heli` start intercepting at two different instants of time, the obligation instance is considered fulfilled when the first agent starts intercepting. In order to capture the intended meaning we need a combination of an obligation to have someone intercepting before the deadline and a prohibition from having more than one agent intercepting the same boat.

Table 1.a shows execution times and number of states explored for the model checking of different scenarios[1]. **cA** and **uB** are the number of coalition agents and unauthorized boats respectively. Columns **nd1** to **nd4** specify whether each norm was included or not (nd1 and nd4 are the two prohibitions that specify norm 1). We verified a `true` safety property which requires the model checker to explore all the reachable states.

Table 1.b illustrates that modelling deadlines, even using Lamport's abstraction [2], has a significant impact on execution times. Increasing the number of agents modelled also has an important effect, but this may be mitigated by exploiting domain symmetries (e.g. all UAVs may be behaviourally equivalent).

5. CONCLUSIONS

We have outlined the syntax and semantics of CÒIR, a normative language that supports the specification of contrary-to-duty, collective and event-driven imperatives with deadlines. We have shown how LTL properties can be verified via the Maude implementation of CÒIR and briefly discussed some of the challenges in model checking such complex normative systems specifications.

Acknowledgments

This research was sponsored by Selex ES.

REFERENCES

[1] M. Clavel et al. *All about Maude-a high-performance logical framework*. Springer, 2007.

[2] L. Lamport. Real-time model checking is really simple. In *Correct Hardware Design and Verification Methods*, pages 162–175. Springer, 2005.

[3] T. J. Norman and C. Reed. A logic of delegation. *Artificial Intelligence*, 174(1):51 – 71, 2010.

[4] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[1] All tests ran on a Intel Core i5 2.7Ghz, 16 GB RAM.