# Generating Various and Consistent Behaviors in Simulations

Benoit Lacroix, Philippe Mathieu and Andras Kemeny

**Abstract** In multi-agent based simulations, providing various and consistent behaviors for the agents is an important issue to produce realistic and valid results. However, it is difficult for the simulations users to manage simultaneously these two elements, especially when the exact influence of each behaviorial parameter remains unknown. We propose in this paper a generic model designed to deal with this issue: easily generate various and consistent behaviors for the agents. The behaviors are described using a normative approach, which allows increasing the variety by introducing violations. The generation engine controls the determinism of the creation process, and a mechanism based on unsupervised learning allows managing the behaviors consistency. The model has been applied to traffic simulation with the driving simulation software used at Renault, SCANeR© II, and experimental results are presented to demonstrate its validity.

## 1 Introduction

A typical application of multi-agent based simulations is the reproduction of real world situations. In such cases, the simulations validity is first assessed by reproducing known situations, before studying how new designs or behaviors influence the results. The variety and the consistency of the behaviors are fundamental in

Benoit Lacroix
Renault, Technical Center for Simulation and LIFL, University of Lille, France
e-mail: benoit.lacroix@renault.com

Philippe Mathieu
LIFL, University of Lille, Cite Scientifique Bat M3, 59655 Villeneuve d'Ascq, France
e-mail: philippe.mathieu@lifl.fr

Andras Kemeny
Renault, Technical Center for Simulation, 1 av. du Golf, 78288 Guyancourt, France
e-mail: andras.kemeny@renault.com

these applications. The variety is often obtained by providing the agents with individual characteristics [3]; the consistency is essential for the results validity, which will be questioned if aberrant behaviors appear. However, these issues are often not specifically considered. A model designed to ease the work of the designers has to consider them and to include different characteristics: provide a high-level representation of the behaviors, allowing abstracting the domain specificities; be flexible and generic to adapt easily to various users needs; and finally, allow users to check that the produced behaviors remain in the limits they wish.

The paper is organized as follows. We introduce in section 2 the different characteristics of the proposed model. The behaviors are described using a normative approach, taking advantage of the descriptive capacities of norms, rather than the more usual prescriptive way typically used in multi-agent systems. The creation of the behaviors is based on nondeterministic principles, and unsupervised classification is used to control the potential violating behaviors and the norms evolution. In section 3 we describe how the model allows producing variety and consistency, and illustrate the genericity of the approach. Section 4 describes the application of the model in the traffic simulation field, and the implementation in the driving simulation software used at Renault, SCANeR© II. Finally, section 5 presents experimental results demonstrating the model validity.

## 2 Description of the Model

### *2.1 Normative Representation of the Behaviors*

In multi-agent based simulations, normative systems are usually used to introduce regulation possibilities in the environment, and to add cooperation and coordination mechanisms [1, 14]. For instance, Electronic Institutions [10] exploit them to regulate agents interactions: the institution describes the conventions governing agents interactions, and the norms assess actions consequences within the scope of the institution. Normative models are applied in various fields: disaster management, market monitoring...

In simulations where the behaviors rely upon many parameters of different kinds (discrete, continuous...), controlling their values and their consistency is a complex issue. The notion of norm, which presents an intuitive description means of the parameters sets, has been proposed to answer it [7]. Norms are used to describe agents behaviors, using the following metaphor: the institution handles parameters limits, and norms behaviors families. The institutional elements are defined as follows.

**Definition 1.** *An Institution is a tuple $\langle P, D_P, P_i, P_e \rangle$ where: $P$ is a finite set of parameters; $D_P = \{d_p, \forall p \in P\}$ is a set of definition domains; $P_i$ is a set of institutional properties; and $P_e$ is a set of environmental properties.*

The institution provides a fixed reference for the norms. It handles a finite set of parameters. A definition domain is associated to each of them, to provide limits

for the parameters. Finally, sets of institutional and environmental values link the institution to its context. Application or domain specificities are taken into account this way.

**Definition 2.** *A Norm is a tuple* $\langle I, P_n, D_{P_n}, \Gamma_P, P_{n_d}, P_{n_i}, P_{n_e} \rangle$ *where: I is the institution the norm refers to;* $P_n \subset P$ *is the subset of parameters associated to the norm;* $D_{P_n} \subset D_P$ *is the subset of definition domains;* $\Gamma_{P_n} = \left\{ \gamma_{p_n} : d_{p_n} \rightarrow \mathbf{R}, \forall p_n \in P_n \right\}$ *is a set of distance functions;* $P_{n_d} = \left\{ p_{n_d}, \forall p_n \in P_n \right\}$ *is a set of default values of the parameters;* $P_{n_i}$ *is a set of institutional properties; and* $P_{n_e}$ *is a set of environmental properties.*

A norm is defined as a subset of the institution parameters, associated to a subset of the definition domains. For each parameter, a distance function is specified, which provides a metric allowing quantifying the final parameter value regarding its original definition domain. A set of default values for the parameters is defined. Finally, the norm handles a set of institutional and environmental properties, which can specialize the institution's ones. Conflicting norms are allowed, several norms can be defined for the same environment, and norms can have non-empty intersections.

**Definition 3.** *A Behavior is a tuple* $\langle N, P_b, V_{P_b} \rangle$ *where: N is a reference to the instantiated norm;* $P_b$ *is a subset of the set of parameters defined in the instantiated norm; and* $V_{P_b}$ *is the set of values associated to the parameters.*

A behavior is the instantiation of a norm. Each element of the behavior is described by a parameter taken from the corresponding norm, and a value associated to this parameter. This value can be taken in or outside the definition domain, but has to remain within the institution's one. A behavior having at least one of its parameters values outside the definition domain specified in the norm is in violation.

## *2.2 Generation engine*

The instantiation from norm to behavior uses a generation engine build on nondeterminism principles, like applied in some displacement models [13]. It does not include domain specific elements, to preserve genericity and flexibility. Its main characteristic is to manage the determinism of the creation process: this way, users are able to guarantee the simulations reproducibility when needed, while allowing the creation of unexpected behaviors in other cases.

Each agent is associated to a finite set $O$ of available objects. All objects in $O$ are balanced with a factor $p_o$ ($\forall o \in O,\ p_o \in [0,1]$ *and* $\sum_{o \in O} p_o = 1$). A deterministic process $d$ is associated to the agent to select the next object. Let $p$ be a random parameter, $p \in\ ]0,1]$ (uniform distribution). At each time step $t$, $O_t \subset O$ is the set of objects which can be selected. Using the probability $1/p$, the agent uses randomly one of the objects in $O_t$, else it uses the deterministic process $d$ to choose it. $p$ itself is randomly chosen with a probability $q$. If $q = 0$, the probability to choose randomly

an object is null: the resulting behavior of the agent is deterministic. If $q = 1$, the object is randomly chosen at each step, and the behavior is purely non-deterministic. When $0 < q < 1$, the randomization level of the process changes depending on the user choice: the number of random drawings increases with $q$.

The engine runs in three steps (Algorithm 1). First, the parameter $q$ is chosen by the user or loaded from the configuration. Then, $p$ is computed at each time step, and used to select the next object $o$. Finally, the agent applies $o$ according to its own factor $p_o$. This allows managing the determinism's level: either follow the provided deterministic process, or easily introduce nondeterminism. The engine is used to instantiate the behaviors. To do so, it is applied at the norm level: $O = P_n$. $O_t = P_b \subset P_n$ is the set of available parameters for the processed Behavior. Finally, the parameter's value $v_{p_b}$ is selected by the model from the corresponding norm's definition domain $d_{p_n}$.

**Algorithm 1** Generation engine

**Require:** $a$ an agent associated to a set of objects $O = \{(o_i, p_{o_i}), i \in I \subset N\}$; $q \in [0,1]$ global randomization parameter; at time $t$: $O_t \subset O$ available actions

$p \leftarrow random(]0,1])$
**if** $p < q$ **then**
  **for all** $o_i \in O_t$ **do**
    **if** $\sum_0^i p_{o_k} < p \leq \sum_0^{i+1} p_{o_k}$ **then**
      $o_{selected} \leftarrow o_i$ {select $o_i$}
      break
    **end if**
  **end for**
**else**
  $o_{selected} \leftarrow$ d( $O_t$ ) {select the object using the deterministic process d}
**end if**
apply( $o_{selected}$, $p_{o_{selected}}$ )

## 2.3 Control mechanism

In order to manage the evolution of agents' behaviors, a control mechanism based on unsupervised learning is used. The data we classify are behaviors, which can be put under a vectorial form (vectors of parameters values). Among the various classical algorithms [5], Kohonen neural networks offer the characteristics we are looking for [6]. The classification is first used to study the agents' behavior evolution during the simulation: they can be classified to check if their behavior remains in the original norm, or matches another one. The second use is to control the evolutions of the norm set. When agents evolve during the simulation, new behaviors emerge. The study of these behaviors can lead to observe new norms (i.e. sets of similar behaviors), which are characterized using the classification mechanism. This can help users to improve their design and analyze the simulations results. Finally, the

network can be trained on real data sets: the norms produced can then be used to recreate behaviors similar to the input data.

To handle these applications, a reference network is trained during the simulation using the instantiated behaviors, before the simulation could modify their characteristics. Only behaviors respecting the norms are used for this training. The study of the behaviors evolution is done by classifying them with the reference network. The evolution of the norm set is observed by training another network using the current behaviors, and comparing the two networks.

## 3 Variety and Consistency

The normative data structure provides two different ways to produce various agents' behaviors. The first method takes advantage of the norms description capabilities: any norms can be defined, each one using a different set of parameters, associated to different definition domains. A large potential of behaviors is thus made available, and very specific behaviors can be created by restricting the domains to single values. Unusual behaviors may be built this way, while keeping full controls over the simulation: the behavior is still the instantiation of a norm, and will never be produced again if this norm is deleted. Note that within the limits of each norm, the generation engine introduces the desired behavioral variety. The second method is based on norms violations. The parameters and associated values are determined during the instantiation, using the generation engine (Algorithm 2): if $q \neq 0$, violating behaviors can appear. This possibility to create violations allows the appearance of unspecified behaviors, which increases the variety.

**Algorithm 2** Instantiating violating behaviors

**Require:** $a$ an agent applying norm $N$; $q \in [0,1]$ the global randomization parameter
**for all** $p_n \in P_N$ **do**
  $p \leftarrow random([0,1])$
  **if** $p < q$ **then** {violation allowed}
    $r \leftarrow random([0,1])$
    **if** $r < p$ **then**
      $v_{p_n} \leftarrow random(D_P)$ {take the value in institution's domain}
    **else**
      $v_{p_n} \leftarrow random(D_N)$ {take the value in norm's domain}
    **end if**
  **else**
    $v_{p_n} \leftarrow random(D_N)$ {no violation allowed: use the norm's domain}
  **end if**
**end for**

As for the behaviors consistency, the limits set with the norms guarantees it if violations are forbidden. To allow reproducible simulations, we only have to store the seeds used by the generation engine. When violations are allowed, the consistency

criteria can no more be guarantied. However, the model definition allows quantifying the violations, using the distance functions specified in the norms. These functions provide quantified values of the deviations, and allows excluding too deviant behaviors.

Finally, once the content of the normative model has been defined, the execution can be done without any modification to the engine. The model is generic, and can be applied in various fields. For instance, if we consider soccer player agents with a single action "shoot" where they have to choose the direction[1], we have an institution, using one parameter *direction* of definition domain $[0, 2\pi]$. Norms can describe players always shooting right or left $D_{P_n} = \{-\pi/4, \pi/4\}$, shooting in front of them ($D_{P_n} = [0, \pi]$)... In artificial economics, market agents are characterized by a direction, a price and a quantity, and apply different norms on the market: zero-intelligent traders, chartists, fundamentalists and speculators can be observed. They can be created with the model, using the same method as presented in section 4.

## 4 Application to Traffic Simulation

In order to illustrate our approach, the model was applied in the context of traffic simulation with the driving simulation software used at Renault, SCANeR© II[2]. SCANeR© IIis dedicated to run a wide range of driving simulators, for various applications: ergonomics of the driver's cab, design, validation of car lightings... In these applications, real humans drive a simulator, immersed in a traffic of autonomous vehicles. In most experiments, the behaviors of the vehicles have to be as realistic as possible, to allow the immersion of the users in the simulation and ensure the validity of the results. However, specific behaviors are sometimes needed, to simulate for instance drunk drivers and study the influence on the reactions of drivers without endangering them.

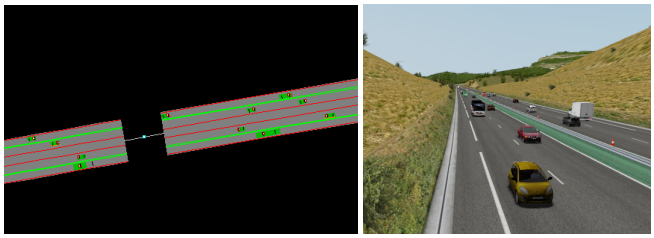

**Fig. 1** The SCANeR© II 2D and 3D visual outputs during the experiment presented in Section 5.

Various traffic management strategies exist, which use different decision models to simulate drivers' behaviors[11]. They often take into account individual charac-

[1] http://www2.lifl.fr/SMAC/projects/cocoa/football.html

[2] http://www.scaner2.com

teristics, including several psychological factors [3]: personality, emotion, motivation and social behavior. In SCANeR© II, the autonomous vehicles use a perception – decision – action architecture as reasoning basis [8]. First, the perception phase identifies the various elements which may interact with the vehicle: roads, lanes, other vehicles, road signs and pedestrians. Then the decision phase is built on three levels: strategic, tactical, and operational. The strategic level plans the itinerary, the tactical level selects the next maneuver to be executed using a finite state automaton, and the operational level computes the acceleration and wheel angles resulting from the chosen maneuver. Finally, the action phase computes the next position, using a dynamic model of the vehicle.

Different pseudo-psychological parameters are used during the decision phase. The "maximal speed" parameter is the maximal acceptable speed for the driver. "safety time" describes the security distance it will adopt, depending on its speed. "overtaking risk" represents the risk a driver will accept to overtake, function of the available gaps with oncoming vehicles. The "speed limit risk" allows it bypassing speed limits, and "observe priority" and "observe signs" are boolean rules regarding the respect of signalization and priorities. These parameters influence the resulting behaviors, and are adapted inputs to the traffic model, so we chose in this work to apply the proposed differentiation model directly on them. The description of the whole set of available parameters constitute the institution: $P = \{$ maximal speed, safety time, overtaking risk, speed limit risk, observe signs, observe priority $\}$; $D_P = \{[0,300],[0,100],[-1,2],[0,10],\{true,false\},\{true,false\}\}$; $P_i$ and $P_e$ are empty sets.

Finally, the model was implemented as an external tool providing input parameters to the traffic simulation model, as presented in Figure 2. It was thus easily introduced in the pre-existing application, as it remains non-intrusive.

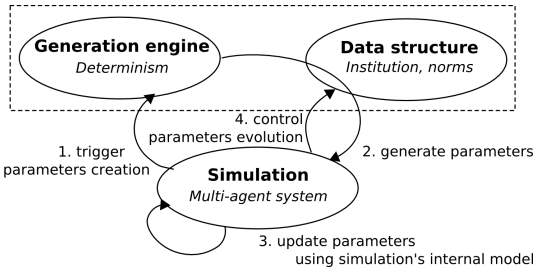


**Fig. 2** Model implementation. The simulation keeps using its internal decision model, and only requires the differentiation model for parameters creation.

## 5 Experimental Results

Based on this institution, three norms were introduced, describing normal, cautious and aggressive drivers (Table 1). They reproduce the behaviors humans are able to distinguish among a simulated traffic [15]. The simulation was done on a database

representing a highway, on a 11 km long section (Figure 1). The vehicles were generated at the beginning of the section, using a traffic demand of 3000 veh/h, during 2h30. The generation function was a uniform distribution. Three detector were placed on the highway, to record vehicles data at kilometer 2.2, 6 and 10.8.

The vehicles were created using three distinct sets of norms. In the first one, *no norms*, all the vehicles are created with the same parameters (the behavioral differentiation model is deactivated). In the second one, *normal only*, only the norm "normal driver" is used. In the third case, *all norms*, the three norms are used, and the norm instantiated to create a new vehicle is chosen with a probabilistic law (cautious 10 %, normal 80 %, aggressive 10 %). After the initial creation by the differentiation model, the traffic model of the application handles all the vehicles.

**Table 1** Norms parameters. The definition domain are truncated normal distribution, presented as following: [minimal value, maximal value], (mean $\mu$, variance $\sigma$)

| parameter | cautious driver | normal driver | aggressive driver |
|---|---|---|---|
| maximal speed | $[90,125]$, $(115,10)$ | $[100,140]$, $(125,10)$ | $[140,160]$, $(150,5)$ |
| safety time | $[1.5,5.0]$, $(2.0,0.5)$ | $[0.8,5.0]$, $(1.5,0.5)$ | $[0.1,1.2]$, $(0.8,0.4)$ |
| overtaking risk | $[-0.5,0.5]$, $(0.0,0.25)$ | $[-0.5,0.5]$, $(0.0,0.25)$ | $[1.0,2.0]$, $(1.5,0.5)$ |
| speed limit risk | $[0.0,1.1]$, $(1.0,0.05)$ | $[0.0,1.1]$, $(1.0,0.05)$ | $[1.0,10.0]$, $(1.5,0.25)$ |

The Figure 3 represents the distributions of vehicles speeds at kilometer 6. When no norm is used, the recorded speeds are either low (70 to 90 km/h, 46 % of the vehicles), or high (130 km/h, 40 % of the vehicles). The left lane remains slow, the right one fast, and we observe few lane changes or overtaking. With one norm, 60 % of the speeds are between 90 and 115 km/h, 30 % between 115 and 140 km/h: the resulting distribution is more balanced. In the last case, the distribution presents a similar shape, widened because of the increased variety of maximal speeds. To evaluate the behaviors consistency, we studied the percentage of each type of drivers on the left and right lane (Figure 4). The set of norms used is the *all norms* case. Most of the aggressive drivers are on the left lane (71 %), when cautious ones stays on the right lane (82 %). In addition, the flow on the left lane represents only 34 % of the total flow. These results are consistent with real world situations. With the addition of norms, an increased variety of behaviors is observed in the simulation, and their consistency is guarantied by the choice of the definition domains.

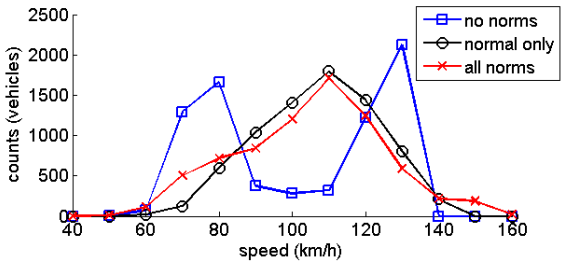


**Fig. 3** Distribution of vehicles speeds at kilometer 6 (total vehicles crossing the detector during the simulation: 7400).

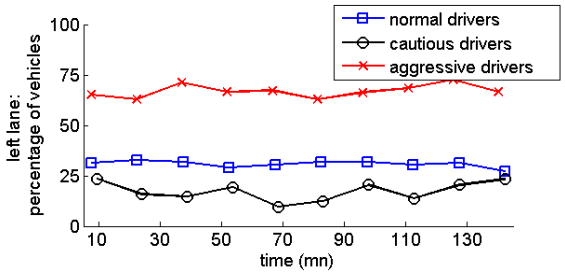


**Fig. 4** Vehicles repartition on the highway lanes, function of their norm.

Different elements can be discussed. Firstly, the values used to define the norms have been chosen empirically: an important improvement would be using calibration with real data, which is currently under work. Secondly, the use of statistical data conceals some of the traffic characteristics. The visual observation of the simulation shows an important increase of behaviors variety (overtaking, speed choices...): we need to introduce indicators quantifying these elements. Finally, violating behaviors were not exploited during these simulations. They will be introduced in further experiments, to simulate for instance loss of control or drunk drivers.

## 6 Related works and Conclusion

The generation of various behaviors in multi-agent systems has been approached from different perspectives. In [9], the authors increase the variety by automatically modifying characteristics of virtual humans in crowds, like clothes or accessories. In [15], virtual personalities are used to improve the behavioral variety in traffic simulation, which contribute to the subjective realism felt by the users. However, these approaches do not handle the issue of behaviors consistency, and the mechanisms remain domains specific. In [12], parameters settings for simulations models are automatically generated using a method based on bayesian networks, without user supervision. However, the consistency issue faced with when dealing with agents behaviors is not taken into account.

Some works have explored the use of normative approaches for traffic simulation. Bou et al. [2] study how traffic control strategies can be improved by extending Electronic Institutions with autonomic capabilities. The system dynamically adapts to maximize the respect of traffic law. In [4], non-normative behaviors are introduced to improve autonomous vehicles behaviors in intersections, by allowing agents to break some of the rules of the road. However, these works focus on the regulation possibilities of norms, when we take advantage of their description capabilities.

In this paper, we have presented a generic model designed to manage the variety and the consistency of agents' behaviors in multi-agents based simulations. These two elements are crucial for the simulations realism, and often not specifically taken into account. The behaviors are described using a normative approach: norms offer an intuitive way to define them, and intrinsically handle the notion of violations.

Their creation is computed using a generation engine managing the determinism of the process. Control capabilities have been added, based on Kohonen neural networks. Their purpose is to check the behaviors deviations against their initial norms, and offer a tool to study the system evolutions. The behavioral variety is achieved by two different ways: with the norms definitions, by creating multiple and/or large definition domains, and by taking advantage of the violation possibilities. In the first case, the consistency is guaranteed within the limits of the norms. In the second case, it has to be checked by the control mechanism, using the possibility to quantify the potential deviations. Finally, the model was applied in the traffic simulation field. It was used to improve the realism of autonomous vehicles behaviors, and experimental results demonstrated the validity of the approach. Introducing various normative behaviors in the traffic increased the traffic dynamicity, while providing consistent drivers behaviors.

## References

1. G. Boella and L. van der Torre. An architecture of a normative system: count-as conditionals, obligations and permissions. In *Int. Conf. AAMAS*, pages 229–231, New-York, USA, 2006.
2. E. Bou, M. López-Sánchez, and J. A. Rodríguez-Aguilar. Adaptation of autonomic electronic institutions through norms and institutional agents. In *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS*, pages 300–319. Springer, 2007.
3. R. Dewar. *Individual Differences*, pages 111–142. Human Factors in Traffic Safety, 2002.
4. A. Doniec, S. Espié, R. Mandiau, and S. Piechowiak. Non-normative behaviour in multi-agent system: Some experiments in traffic simulation. In *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, pages 30–36, Hong Kong, China, 2006.
5. S. I. Gallant. *Neural Network Learning and Expert Systems*. MIT Press, 1993.
6. T. Kohonen. *Self-Organizing Maps*. Springer, 1995.
7. B. Lacroix, P. Mathieu, and A. Kemeny. A normative model for behavioral differentiation. In *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, pages 96–99, Sydney, Australia, 2008.
8. B. Lacroix, P. Mathieu, V. Rouelle, J. Chaplier, G. Gallée, and A. Kemeny. Towards traffic generation with individual driver behavior model based vehicles. In *Driving Simulation Conference North America*, pages 144–154, Iowa City, USA, 2007.
9. J. Maim, B. Yersin, and D. Thalmann. Unique instances for crowds. *Computer Graphics and Applications*, 2008.
10. P. Noriega. *Agent mediated auctions: The Fishmarket Metaphor*. PhD thesis, Univ. de Barcelona, 1997.
11. J. Olstam. Simulation of rural road traffic for driving simulators. In *Transportation Research Board*, Whashigton, USA, 2005.
12. R. Pavón, F. Díaz, and V. Luzón. A model for parameter setting based on bayesian networks. *Engineering Applications of Artificial Intelligence*, 21:14–25, 2008.
13. C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, pages 763–782, San Francisco, California, 1999.
14. J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *J. of Autonomous Agents and Multi-Agent Systems*, 11:307–360, 2005.
15. S. Wright, N. J. Ward, and A. G. Cohn. Enhanced presence in driving simulators using autonomous traffic with virtual personalities. *Presence*, 11:578–590, 2002.