7 A SUMO Extension for Norm based Traffic Control Systems

*Jetze Baumfalk, Barend Poot, Bas Testerink and Mehdi Dastani;
Department of Information and Computing Sciences, Universiteit Utrecht, The Netherlands*

{J.T.Baumfalk,B.W.A.Poot,B.J.G.Testerink,M.M.Dastani}@uu.nl

Abstract

Autonomous vehicles will most likely participate in traffic in the near future. The advent of autonomous vehicles allows us to explore innovative ideas for traffic control such as norm based control systems. A norm is a violable rule that describes correct behavior. Norm based traffic control systems monitor traffic and effectuate sanctions in case vehicles violate norms. In this paper we present an extension of SUMO that enables the user to apply norm based control systems to traffic simulations. In our extension, vehicles are capable of making an autonomous decision on whether to comply with the norms or not. We provide a description of the extension, a summary on its implementation and an experimental evaluation.

Keywords: Normative systems, distributed control.

7.1 Introduction

In the near future, smart roads with autonomous vehicles will become prevalent. Vehicles will no longer have a driver, but will drive themselves and communicate with the infrastructure and other vehicles. This allows, and gives rise to the need, for the investigation of new traffic control measures [11]. Traffic control systems can exert some level of influence on vehicles in order to improve traffic flow and safety. In this paper, we shall focus on control systems where the infrastructure instructs vehicles on how they ought to behave.

Current control techniques include for instance ramp metering where a traffic light, or adaptive speed limit displays, can be used in order to control traffic flows. However, with new autonomous vehicles (see [14],[15],[16]) that have the ability to pursue goals, follow regulations, and perform an array of actions, we can give more individualized directives. We do not necessarily have to rely on drivers to observe road side units, but can communicate directly with vehicles, since the vehicles can communicate with their environment, reason to decide which actions to perform, and execute their decided actions. In the use case shown in this paper we can for instance assign target velocities to individual vehicles. Just like current traffic regulations, it is possible that vehicles violate these regulations by speeding or driving on an emergency lane in order to arrive at their destination at an earlier time. We observe however that microsimulation platforms such as SUMO do not currently allow for vehicles to reason about regulations, the consequences of violations and their personal preferences.

In this paper we present the implementation of a new traffic control system for SUMO. This control system is based on the notion of norms and is hence referred to as a norm based

control system. Norms can be seen as violable rules that specify desirable behavior. Norm based control systems are a proposed technique for solving control problems where the subjects of the regulations are capable of making autonomous decisions on whether to violate the regulations. They lend themselves very well for the cause of traffic control. A norm based traffic control system is a more natural way of modeling roads with different properties and regulations [13]. We also provide a new vehicle driver model so that vehicles can reason about norms. Our implementation is a plug-and-play extension for SUMO. This allows researchers to use our extension out of the box.

We illustrate our framework by applying it to the common example of merging traffic streams. For a schematic overview, see Figure 7-1 where two traffic streams have to be merged together. There is one main traffic stream and one secondary traffic stream that joins the main stream, resulting into a single output stream. The goal is to make optimal use of the output capacity of the network whilst not causing unnecessary traffic jams for the secondary traffic stream or compromising safety. A solid analysis of this scenario can be found in [11]. Our approach is to use a norm based control system that assigns individual norms to agents. In particular, agents are obligated to move or stay on a lane and/or adopt a certain target speed until they are released of this obligation.

Figure 7-11. Example setting where two traffic streams must merge

Our paper is structured as follows. We first give a brief introduction in the field of norm based control systems. We will then show how the norm based control system for SUMO is designed, and discuss an application. Following that, we describe how vehicles can reason about the norms that they receive. In the next section we evaluate our extension through a series of experiments that highlight different aspects of our contribution. After that, we briefly look at related work and see how our approach differs. Finally, we summarize the main points and contributions of our work.

7.2 Norm Based Control Systems

The theory of norm based control systems originates from the Multi-Agent Systems (MAS) community [17]. In the MAS paradigm, the main concepts and abstractions are the environment, agents, norms, and control system. The environment is the system in which agents can act. For our purposes we see SUMO without driver models as the environment. The agents in a MAS are autonomous entities that exhibit their own behavior. We see the drivers of vehicles as the agents in a traffic MAS. Norms specify how agents ought to behave under what circumstances. For instance, traffic regulations could be seen as norms. The control system monitors agents' behaviors and the environment, and checks whether norms are violated. If a norm violation is detected, then the control system can impose a sanction to the violating agent. These sanctions are imposed on agents in order to discourage the agent to violate any traffic regulations. We refer to the agents that operate the vehicles when we refer to vehicles throughout this paper.

Background on Norm Based Control Systems

Multi-agent systems consist of autonomous interacting agents whose behavior is not always predictable or controllable. In addition to reactively responding to environmental changes, autonomous agents are assumed to have their own objectives (goals) for which they proactively initiate actions in order to achieve them. The behavior of individual agents needs to be controlled and coordinated in order to ensure the desirable properties at the system level [17]. One approach is to design and implement hard constraints at both individual agent and multi-agent system levels. This approach may not always be desirable or feasible, since limiting the autonomy of individual agents can be costly or even impossible. For instance, consider future traffic on smart roads as a multi-agent system where vehicle drivers (autonomous cars) are modeled as individual agents and the road infrastructure is modeled as the multi-agent system environment. In such a scenario, it is undesirable, if not impossible, to fully control and determine the behavior of all individual autonomous cars, as these vehicles are designed to make their own decisions.

Norm based control systems are widely proposed as an effective mechanism to control and coordinate the behavior of individual agents [4]. In norm based control systems, norms are considered as specifying the standards of behavior that can be used to govern the interaction between autonomous agents (cf. [10],[5],[2]). Across various theories and frameworks there is no general consensus on what a norm is. In this paper we use the term norm as a reference to both norm schemes and norm instantiations. With the concept norm scheme we refer to the specification of the circumstances after which a specific agent is obliged to achieve a system state, and the sanction that will be imposed should this obligation not be met before a certain deadline (cf. [10]). With the concept norm instance we refer to a specific obligation and sanction that holds for a specific agent. Norms can take the form of an obligation, prohibition or permission, but the scope of this paper concerns only obligation norms. The application of norms in multi-agent systems, called norm based multi-agent systems, requires continuous monitoring of the behavior of individual agents, evaluation of their behavior with respect to the specified norms, and assurance that norm violating agents are sanctioned. This approach maintains the agents' autonomy and can still promote desirable behavior. We observe that traffic regulations fit the normative approach. For instance one can represent the regulation "if you approach a ramp, then you ought to have a safe speed for merging into the traffic stream" as a norm.

Finally, norm instantiations in norm based control systems are announced to autonomous agents, just like drivers are assumed to be aware of traffic regulations. Norm awareness allows an agent to reason and decide whether or not to comply with the norms (cf.[1],[7]). Norm awareness is crucial for future traffic on smart roads as autonomous vehicles need to know the consequences of norm violations before deciding whether or not to comply with the norms.

A Norm Based Control System for SUMO

A norm based traffic control system monitors vehicle behavior and reacts to it. In our extension of SUMO, the monitoring functionality of traffic control systems serves two purposes. The first purpose is to detect violations of norms, such as speeding, tailgating or driving on a priority lane without a permit. If such a violation is detected, then a sanction coupled with this violation will be issued towards the violating vehicle. The second purpose of

monitoring is to issue new norms. If the traffic control system continues to observe situations where either the throughput or safety of vehicles declines, then a norm might be issued to improve the situation. This norm might be global, or tailored to a specific vehicle. This allows the control system to be very adaptive to traffic dynamics. Furthermore, the severity of a sanction might be increased if the coupled violation either occurs an excessive amount of times, or seems to be the cause of problematic situations. In our framework for norm based traffic control, both autonomous vehicles and the control system show adaptive behavior towards the ever-changing traffic flow and enable the system to cope with difficult and dynamic traffic situations. An improvement might also be possible in common situations like ramp-merging. In this scenario, norms issued by the traffic control system are aiding in improving the throughput of vehicles.

The structure of the presented norm based control system for SUMO is depicted in Figure 7-2. As stated before, we use the native SUMO application as the environment. However, we do not use SUMO's driver models. Instead we communicate commands from our own agent model to the vehicles. We have also implemented our own sensor business logic. These sensors are connected to lane area detectors as implemented in SUMO. The communication between the extension software and SUMO is provided by the TraaS library. We use and provide a new version of TraaS. In addition to improved performances, the new version has some extra functionalities that are needed for our extension.

The software in our extension is composed mainly of the agent models and the control system. The control system monitors vehicles through its sensors. It then checks the norms to determine whether new obligations hold for specific agents (if so, then they are also communicated to said agents) and/or whether

Figure 7-2. Overview of the presented extension.

norms are violated. The software executes in lockstep with SUMO, i.e. each simulation tick in SUMO is also a tick in the extension.

We chose to provide our work as an extension to SUMO, rather than modifying the source code of SUMO directly. There are several reasons for this. First of all, vehicles in SUMO are goal directed only in a limited way. Their goal is to follow a certain route, as opposed to having a specific location as the destination. Second, SUMO agents are preprogrammed to follow a specific route. They only respond reactively to their environment, instead of deliberating on what action would best suit them. Third of all, vehicles in SUMO are not inherently capable of deciding to break the rules. For example, they always stop for a red light and they obey to the right of way. This is because vehicles in SUMO move according to specific car-following and lane-changing models and these models do not allow

reasoning about norms for action decision. The car-following model is not created with norm aware agents in mind. The most commonly used car-following model made by Stefan Krauss is designed purely to create realistic traffic flows in general, since in most traffic simulations individual movement on the microscopic level is not interesting [6]. In contrast, we aim at designing futuristic autonomous cars with a more fine grained sense of control and most importantly, the ability to violate norms.

There are also pragmatic reasons to propose a SUMO extension rather than altering its code. For example, we can now support multiple versions of SUMO, starting from SUMO 0.20 and upwards. This also makes our extension more accessible to users, since it eliminates the need for users to recompile SUMO before they can use the framework. Our extension is thus usable out of the box.

The framework is developed in Java since most norm-based autonomous agent frameworks are written in Java and Java programs are easy to use on a variety of platforms. We have designed the framework using the Model View Controller pattern, as can be seen in Figure 7-3. A multi-agent system can be specified in XML and is converted to a MAS-model. We furthermore use TraaS to retrieve the simulation state of SUMO. These models can be manipulated by the controller software (note that the controller is a control component in the software engineering sense; it is not the same as the conceptual control system for traffic). The agent framework simulates our driver models. The view of the system allows the user to see the state of the MAS side of the simulation. This decomposition allows other researchers to easily create their own front end by changing the view implementation. It is also possible to create a new data format for scenarios by changing the data model implementation, or to change the simulation package by providing a different simulation model implementation. Each part can be changed without the need to change other parts of the framework.

Figure 7-3. The Model-View-Controller structure.

7.3 Application of Norm Based Traffic Control in SUMO

In our use-case scenario we illustrate a fairly simple norm based control system. On the control system side it is calculated when individual vehicles would arrive on the merge point of the traffic streams. The algorithm used is described in [12] and returns an ordering of vehicles. Next, the target velocity is calculated for each individual vehicle that ensures that it a) crosses the merge point at least two seconds after the vehicle that will cross the point before it, and b) the maximum safe velocity is still maintained. In case the main stream road has two lanes, then a vehicle can also instead be obliged to move to the left lane. This happens when target velocity of a vehicle on the main road is below a predefined threshold. The sanctions of violating norms are captured by a low or high fine.

On the agent side of the system we model vehicles as individual agents with a target arrival time. Aside from this, an agent has a personal profile that specifies how desirable it is for the agent to be on time and how undesirable it is for the agent to receive a sanction. This can lead to different behaviors such as an affluent agent in a hurry opting to take a high fine by

increasing its speed, in contrast to an equally hasty but poor agent not being able to afford the fine which then obeys its assigned target speed. Our main goal is to model future traffic on smart roads as norm based multi-agent systems and simulate such traffic scenarios in SUMO. In order to achieve this objective, we extended SUMO to simulate i) norm-aware autonomous cars, and ii) norm based traffic control systems consisting of monitor and control mechanisms.

Scenario details

In Figure 7-4, a schematic representation of the aforementioned ramp merging scenario is given. Triangles are vehicles that travel in the direction towards they point. White vehicles are the vehicles that have not yet received their obligation from the control system. In this case those are vehicles A, B, C and D. On the road there are lane sensors (s_1 to s_5) which can detect the status of vehicles that are residing on them. For the scenario to work correctly, it is necessary that either the sensors are sufficiently long, or the vehicles sufficiently slow, so that no vehicle can pass by undetected. The sensors should also be placed at a distance far enough from the merge point m , so that vehicles have enough time to comply to the obligation before the deadline. There are two important points on the road, point m where the two roads merge, and point e where the vehicles exit the scenario. Distances d_A and d_C are agent's A and C's distances to m , and d_{exit} is the distance from the merge point to the exit point, and d_{safe} is the distance between vehicles that is deemed safe, the minimal gap. Ideally A and C traverse d_A and d_C s.t. they arrive at m with a distance d_{safe} and can accelerate to their maximum speed within the distance d_{exit} .

Figure 7-4. Use case with two lanes.

Monitoring happens through interpreting the observations of sensors. In the case of SUMO we use lane detectors that can sense the vehicles that driving along the area they cover. Specifically, each sensor can detect the identity, velocity and position of each vehicle on the sensor's area. Further parameters such as the maximum velocity, acceleration and deceleration capabilities can be assumed within reasonable margins. The control system uses the sensor data to check given norm schemes whether new norms are instantiated, as well as checking whether instantiated norms are violated. If a new norm is instantiated, then the subjected

vehicle is notified of the obligation that it has to fulfill, and the associated sanction that will be imposed if the obligation is not fulfilled.

The standard traffic rule is that the stream that originates on the main road has priority over the ramp road's stream. However, if the main road is busy, this may lead to large traffic jams on the secondary road. Therefore the organisation uses the traffic data from sensors 1 to 3 calculate how fast vehicles must go so that on the merge point the traffic streams smoothly merge together. More specifically, the organisation tells vehicles passing sensor 1 to stay on the left lane. Sensor 2 and 3 are used by the organisation to coordinate the scheduling of vehicles on the merge point. If a vehicle passing sensor 2 has to slow down too much, i.e. to a velocity less than the threshold, then it receives the obligation to move to the left lane.

In order to not overcomplicate the scenario we decided to simplify some aspects of the control system. The sanction that an agent can receive for violating a norm is modelled by either a low or high fine. In general the set of possible sanctions is S , which for our scenario is: $S = \{fine_{low}, fine_{high}\}$. The set of possible obligations is O . An obligation that a vehicle can receive is an obligation to be on the left or right lane of the main road at a certain target velocity. For instance (l_{right}, v_{10}) is read as the obligation to be on the right lane at 10 m/s. For our scenario the possible obligations are: $O = \{l_{left}, l_{right}\} \times \{v_x | x \in \mathbb{R}\}$. A pair of an obligation with a sanction is called a norm instantiation. The set of all norm instantiations is denoted with $N = O \times S$. For each simulation step the new norm instantiations are created. Recall that the specification of how norms are instantiated is referred to as a norm scheme.

Example Norm Scheme

For our scenario, an organization instantiates a norm scheme into agent specific norms, so that two vehicles cannot arrive on the merge point at the same time, if they comply with the norm. This will cause vehicles to slow down considerably on the main road. If they have to slow down too much then they will be obliged to move to the left lane which is assumed to be more free flowing. The exact explanation of this norm is provided in Appendix A.

We also have another norm scheme that obliges vehicles that enter on the left lane of the main road to stay on their lane at a preset maximum velocity v_{MAX} . We shall illustrate the different aspects of a norm scheme according to this scheme's pseudo code that is given in Table 7-1. We assume that there is a current set of norm instantiations NI, a function *sanction* that given an agent and fine issues the fine for that agent, and a function *read* that returns the vehicles on a sensor's area that have not been seen before by that sensor. Though not a forced pattern, we do encourage future users of our extension to use the same code structure as in Table 7-1. Every norm scheme has a specification of when norm instances are created, when they are retracted, and when a sanction should be issued.

The code in Table 7-1 is executed after every simulation tick. We begin with creating new norm instances (lines 0-3). In this case sensor 1 is read. In Figure 7-3 it can be seen that this sensor detects all vehicles that enter the scenario on the left lane of the road. Hence, we give each agent on the left lane the obligation to stay on the left lane, and also obtain a preset maximum velocity to ensure that the flow stays high (line 2). We then continue by checking sensor 4 (lines 4-10). Vehicles with instantiations of this norm that pass sensor 4 fulfilled their obligation to stay on the left lane. However, if their velocity is not optimal, then they get a low fine anyway (lines 6-8). After passing this sensor the vehicles are relieved of their obligation (line 9). The same holds for sensor 5 (line 16). However, if a vehicle passes sensor 5

then it means that it switched to the right lane. Hence, each vehicle that has received an obligation to stay left but passes sensor 5 is fined (line 14).

Table 7-6. Pseudo code for the stay-on-lane norm scheme.

```

0:   $L \leftarrow \text{read}(s_1)$ 
1:  for each  $\text{agent} \in L$  do {
2:       $NI \leftarrow NI \cup \{(\text{agent}, ((l_{\text{left}}, v_{\text{MAX}}), \text{fine}_{\text{low}}))\}$ 
3:  }
4:   $L \leftarrow \text{read}(s_4)$ 
5:  for each  $\text{agent} \in L$  do {
6:      if  $(\text{agent}, ((l_{\text{left}}, v_{\text{MAX}}), \text{fine}_{\text{low}})) \in NI \ \& \ \text{agent}.s < v_{\text{MAX}}$  then {
7:           $\text{sanction}(\text{agent}, \text{fine}_{\text{low}})$ 
8:      }
9:       $NI \leftarrow NI \setminus \{(\text{agent}, ((l_{\text{left}}, v_{\text{MAX}}), \text{fine}_{\text{low}}))\}$ 
10: }
11:  $L \leftarrow \text{read}(s_5)$ 
12: for each  $\text{agent} \in L$  do {
13:     if  $(\text{agent}, ((l_{\text{left}}, v_{\text{MAX}}), \text{fine}_{\text{low}})) \in NI$  then {
14:          $\text{sanction}(\text{agent}, \text{fine}_{\text{low}})$ 
15:     }
16:      $NI \leftarrow NI \setminus \{(\text{agent}, ((l_{\text{left}}, v_{\text{MAX}}), \text{fine}_{\text{low}}))\}$ 
17: }

```

7.4 Norm-Aware Vehicle

Our goal is to build autonomous norm aware vehicles that operate on smart roads where vehicles' behaviors are automatically monitored, evaluated with respect to some traffic norms, and possibly sanctioned. This means that individual vehicles have their own objectives (e.g., destination, arrival time, travel cost, etc.) and are able to deliberate on how to achieve their objectives. This implies that a vehicle can choose to obey/violate a norm, when this helps with achieving its objectives.

A Norm-Aware Driver Model

In Figure 7-5 the general Sense-Reason-Act cycle of an autonomous vehicle is illustrated. In order to model norm awareness we deviate in our scenario from the standard driver models that SUMO provides [6]. A vehicle perceives its road environment by processing the information it receives from SUMO. That information also includes which obligations are newly instantiated or retracted. Using its knowledge of the the road network, the vehicle then estimates the utility of each of its actions by balancing between the value of achieving its objectives and possible sanctions that will incur if it performs the action. For example, the

Figure 7-5 The Sense-Reason-Act cycle of an agent

arrival time and additional fine payments are used to calculate the utility for each action. The action with the highest utility is chosen, using a priority based tie-break mechanism for ties. One autonomous vehicle differs from another through its personal profile. A personal profile states not only its goals and current norm instances, but also how severe sanctions are for the vehicle and how important it is to arrive on time at the goal location. Since each driver profile is unique, different emergent behavior can be observed between different drivers. For instance, a wealthy driver can at times choose to violate speeding norms, since it can afford the fines. However, a poor driver cannot afford the fines, so it will not break the speed limit.

The required components and concepts for a norm aware agent in our system are:

- A set of actions to choose from.
- A description of the physical state of the driver's vehicle.
- A function that estimates the next physical state given a physical state and an action.
- A personal profile that contains:
 - o A goal location and time.
 - o A function to calculate how good an expected arrival is compared to the goal time.
 - o A function that grades how bad sanctions are.
 - o The current norm instances of the agent.
- A function, called the obligation distance measure, that given a physical state of the vehicle and a norm instance, returns how likely it is that the sanction will be imposed upon the driver.
- A utility function that given a possible actions, the current physical state and the personal profile returns the expected utility of the action.
- An action selection function that picks the best action given the current physical state, and personal profile.

Note that the personal profile and the obligation distance measure are not fixed. While it is recommended to let the functions have certain properties (e.g. the grade functions should be monotonic), our framework allows for arbitrary implementations of these functions.

In the next section we give examples of each of these components. Before a simulation tick in SUMO each agent determines what the next best action is by calculating for each action the utility. This is done by determining for each action the next expected physical state and then look at how likely sanctions will be incurred in the future given that next state. If a norm instantiation's sanction is severe, then moving away from its instantiated obligation will decrease the utility. Also if

Figure 7-6 An UML overview of the Agent structure

the target arrival time is less likely to be met, then the utility also decreases.

Norm Aware Driver Model for SUMO

An UML overview of the agent structure is shown in Figure 7-6. Agents are an extension of physical objects since they occupy a certain space and have a certain position in the world. A Basic Agent is a prototype AgentProfile. The agent profile is specified by the SactionGrade and ArrivalTimeGrade functions which are the sanction and arrival time appraisal functions discussed in section 1.5.2. In the doAction function the obligation distance measure is used to calculate the utility for each individual action.

In this manner, the agent profile of the agent together with the current state decides what action the agent will choose towards achieving its goal. These design choices were made to model the agents in such a way that new agent profiles can easily be created to model norm-aware human or driverless autonomous vehicles, while still keeping the agents as simple as possible.

With each deliberation cycle agents can pick the an action most suited to their goals and the current state, one of the actions listed by the AgentAction enumeration in Figure 7-6. The displayed *changeRoad* action is not used in the experiments, but is used to change a road in the current route of the agent. In our current implementation this action is used when the agent receives information about traffic jams at locations he still plans to visit. The agent can then plan a new route, make an estimation to check if the new route might be faster, and change its current route. This is one feature that show the pro-active attitude that the TrafficMAS agents possess.

7.5 Application of Norm Aware Driver Models

In this section we shall go into detail how norm awareness is achieved in our example scenario.

Agent Specification

For our application the set of actions are de-/accelerating a certain amount (denoted as a_x) or switching a lane to the left or right (denoted as l_{left}, l_{right}). More specifically, the set of actions A that a vehicle driver can possibly perform are:

$$A = \{a_x | x \in \{0, 0.01, -0.01, 0.1, -0.1, 1, -1, 5, -5, 10, -10, 20, -20, 50, -50\}\} \cup \{l_{left}, l_{right}\}$$

A vehicle's physical state s is specified by $\langle v_{max}, v, l, d \rangle$, where v_{max} is the maximum velocity, v is the current velocity, l is the current lane, and d is the distance from the startpoint of that lane. Finally, B is used to denote the set of all possible physical states.

An agent's personal profile p is specified by $\langle g_s, g_t, n \rangle$, where $g_s: S \rightarrow \mathbb{R}$ is the sanction grading function that returns how important it is to avoid a given sanction from S , $g_t: \mathbb{N} \rightarrow \mathbb{R}$ is the arrival time grading function that returns how good it is to exit the scenario at a given time, $n \subseteq N$ are the current norm instantiations. With P we denote the set of all possible personal profiles.

Agents need to reason about the future in order to determine the utility of actions. We assume that there is a given function $f: B \rightarrow \mathbb{N}$ that returns given a physical state an expected arrival time. This function reflects for instance the GPS planning tools that vehicles have

available. This function is uniform for all agents, but can be parameterized in the future in order to make more optimistic/pessimistic agents. We also assume agents have a local action effect function $e: B \times A \rightarrow B$ that returns given a physical state and an action the next expected physical state. For instance: if the current velocity is 20 m/s and a vehicle accelerates by 5 as an action, then it expects for the next simulation tick to be at 25 m/s if this is possible within its acceleration capabilities.

Due to physical constraints, it may not be the case to adhere to obligations immediately after they are received. Hence, an agent must be able to deliberate whether an action will bring it closer or further away from fulfilling an obligation. To this end they use an obligation distance measure $\delta: B \times O \rightarrow \mathbb{R}$ which returns given a physical state and an obligation, a positive expectancy of whether the agent can fulfill the obligation in time before it is sanctioned. The precise definition of this function depends on the possible obligations and agent specifics in an application. However, a high distance should mean that it is likely that the sanction will be incurred in the future whereas a distance of zero should indicate that the current state fulfills the obligation.

In the case of our scenario the factors that an agent considers to compute δ are the time it takes to fulfill the obligation when doing so as soon as possible (based on the vehicles acceleration, local traffic data etc), the current time and the latest expected time that the sanction will be issued if the obligation is not met. We have implemented vehicles in such a way that they expect that the control system will check whether an obligation is fulfilled somewhere between the current time t and the current expected exit time t_{exit} for the vehicle. The minimal amount of steps needed to adhere to the norm is denoted δ_t . For instance, if a vehicle at time step t is being instructed to drive 25 m/s, and can accelerate to this speed in minimally 3 time units, then $\delta_t = 3$. If we need zero steps to adhere to the norm, the distance is zero. Otherwise the distance proportionally moves to 1 given the current time. If the control system will check obligation fulfillment before the agent can achieve adherence (i.e. $t_{\text{exit}} - t < \delta_t$), then δ should be 1.

The exact function given the current time t , the expected exit time t_{exit} and the minimal number of steps needed for adherence δ_t is:

$$\delta(\langle v_{\text{max}}, v, l, d \rangle, (l_o, v_o)) = \frac{\min(\delta_t, t_{\text{exit}} - t)}{t_{\text{exit}} - t}$$

Note that this means that the agents expect the control system to issue a sanction if an obligation is not met between now and $t_{\text{exit}} - t$ time units after that. This distance measure can be modified easily in our framework.

Agents use an action selection function $\alpha: B \times P \rightarrow A$ that given a physical state and agent profile returns an action. This selection is based on the utilities of actions. For each action it is simulated what the new expected arrival time and physical state will be. The result is used to calculate how good a new state is given the distance to fulfilling obligations (and thereby avoiding sanctions) and also how good the new arrival time is. The utility function $u: B \times P \times A \rightarrow \mathbb{R}$ is:

$$u(b, \langle g_s, g_t, n \rangle, a) = g_t(f(b')) + \sum_{(o,s) \in n} (\delta(b', o) \cdot g_s(s))$$

where $b' = e(b, a)$.

Example Utility Calculations

For example, suppose we have two agents, a poor one and an affluent one in an identical situation. They currently drive 20m/s, their maximum speed is 30 m/s, they can accelerate or decelerate with 10m/s and their travel distance is 1080 meters. Both are in a hurry, so their g_t is defined as $g_t(time) = bestTime/time$.

Here, *bestTime* is defined by the minimal travel time, i.e. the time it would take the agents to travel the distance if they could go their maximum speed all the time. In this case, $bestTime = \frac{1080}{30} = 36$.

However, the road the agents travel on has a speed norm, with the maximum speed being 10 m/s. Not obliging to this norm gives a $fine_{high}$. The

poor agent cannot afford this fine, so it has $g_s(fine_{High}) = -20$. The affluent agent can easily afford this fine, so it has $g_s(fine_{High}) = -0.2$. Suppose for this example, that agents only can take the actions a_0, a_{10} and a_{-10} . In Table 7-2, we see the utilities for each of these actions and both agents. Here we see that the highest rewarded action for the poor agent is to indeed oblige to the norm, since it cannot afford the fine. The affluent agent is in a position to not decrease its speed to oblige to the norm, since it can afford the fine. In fact, it will even increase its speed, since it then maximizes its time grade, yielding a higher utility.

Table 7-1. Example of a poor and affluent agent.

	a_0	a_{10}	a_{-10}
Speed after action	20m/s	30m/s	10 m/s
Norm speed	10 m/s	10 m/s	10m/s
Exit time with this speed	54	36	108
g_t	0,67	1,00	0,33
steps needed to oblige to norm	1	2	0
δ	0,02	0,06	0,00
u agent poor	0,30	-0,11	0,33
u agent rich	0,66	0.99	0.33

7.6 Experimental Evaluation

We tested the performance of our normative agent based traffic approach using three experiments. In each of these experiments variations of the ramp-merging scenario as explained in section 7.3 is used. The first experiment considers a ramp-merging scenario where the main road consists of a single lane, while on the second and third experiment the main road has two lanes. In the second experiment this second lane is accessible for all agents, but in the third experiment this lane is marked as an "emergency only" lane.

The experiments were set up as follows. Each experiment has a length of one hour (3600 ticks). The spawn rate shown in the tables of the experiments is defined as the chance of an agent spawning every tick. If there is not enough room to spawn an agent at a certain time, then SUMO puts the agent on hold and spawns it at the earliest possible time there is space available. The maximum speed at the merge point, v_{max} , was set to 80 km/h. Furthermore, all three experiments consists of comparing two scenario's, both are run one hundred times. The values displayed in the tables are the averages over those hundred runs. The Throughput is

defined as the number of vehicles leaving the simulation every tick. Average speed is the average speed over all runs in m/s, and finally the Average gap is the average distance between two cars in meters. Also defined for each experiment is the maximum (expected) throughput, this is the expected throughput if each vehicle could keep driving its maximum speed throughout the scenario and can be calculated by the following formula: $\text{throughput}_{\max} = 60 * p$, where p is the total chance of a car entering the simulation on that tick. Since SUMO does not spawn vehicles when space is too limited, and this is the case when traffic jams occur, we believe a significant difference in throughput is a sufficient validation of improvement.

Experiment 1: SUMO and TrafficMAS

The first experiment illustrates the distinction in behavior between the default SUMO vehicles and the norm-aware agents implemented in the TrafficMAS extension. A single norm based traffic control system observes the vehicles in the simulation and communicates tailored norm instantiations to each vehicle. In this experiment a norm instantiation is simplified to only a target velocity since there is no choice of lanes on the main road. The expected result is that these personalized norms will result in a higher average velocity and a better throughput of vehicles since traffic

jams will be prevented. In this scenario a classic ramp-merging situation is implemented, where both the main road and ramp consist of a single lane. The spawn rate of the vehicle input stream will be slightly higher on the main road to resemble a realistic traffic situation. In the *TrafficMAS* scenario three sensors are placed on the road, one on the main road, one on the ramp and a control sensor on the output road. The traffic control system can only communicate (instantiate norms, remove norms or apply sanctions) when the vehicles are driving on said sensors. In the *SUMO* scenario the main road has priority over the ramp, comparable to an authentic merging situation.

Table 7-3: results for experiment 1.

	SUMO	TrafficMAS
<i>Main road Spawnrate</i>	20%	20%
<i>Ramp Spawnrate</i>	15%	15%
<i>Throughput</i>	16,16	21,01
<i>Average Speed</i>	3	20,97
<i>Average Gap</i>	13,81	101,821
<i>Max throughput</i>	21	21
<i>Throughput %</i>	76,95%	100,05%

Figure 7-7: Screenshot depicting the difference in performance in experiment 1. The top scenario uses our framework and merge norm. The bottom scenario uses the default SUMO driver models.

As is clear from the results in table 7-2, there is an increase in both throughput, average speed and the average amount of space between the vehicles. This is the case since in the SUMO scenario a traffic jam instantly forms on the ramp, because of the relatively high density of cars on the main road. These results confirm our expectation of coordination by a norm-based traffic control system improving on classic ramp-merging scenarios. Note that the *throughput* % value exceeds a hundred percent, this is possible because the spawnrate is probability based and thus can exceed the maximum expected throughput.

Table 7-4: results for experiment 2

Experiment 2: Simple norms and Advanced Norms

This experiment is a comparison which demonstrates the distinction between norm-based traffic control systems using simple or advanced norms. Each scenario is observed by a traffic control system, where the control system in the *SingleNorm* scenario observes and controls the same norm as in experiment 1. In the *AdvancedNorm* scenario the control system effectuates a more advanced norm, advanced in the sense that the obligation that the agent receives can consist of different goals instead of just a target velocity. In this case the other obligation that can be issued is for the agent to change lanes in order to relieve the rightmost lane traffic and prevent congestion. This lane change obligation will be given to an agent when its calculated velocity on the merge point is below a certain threshold. For this experiment the threshold was set to $\frac{1}{2}v_{max}$. Our expectation is that in this multi lane scenario, the control system with the

	SingleNorm	AdvancedNorm
Main road Spawnrate	30%	30%
Ramp Spawnrate	20%	20%
Throughput	20,38	29,91
Average Speed	3,31	14,91
Average Gap	14,2	61,49
Max throughput	30	30
Max throughput %	67,93%	99,7%

advanced norm can successfully cope with a higher input stream of vehicles; the traffic control system which effectuates the simple norm, will not.

The setup for the *SingleNorm* scenario is almost an exact copy of the *TrafficMAS* scenario in experiment 1, the two distinctions are that the main road has two lanes instead of one and the input stream of vehicles of both roads are increased. The *AdvancedNorm* scenario also implements extra sensors on the second lane, but is exactly the same in every other aspect.

As can be observed from the results, the simple norm cannot cope with the increased spawn rate of vehicles in this scenario. The average speed is has diminished severely, as well as the average gap between vehicles. This means congestion is abundant in the *SimpleNorm* scenario. However, the *AdvancedNorm* seems to cope very well with the increased input stream of vehicles. In this scenario the throughput approximates the maximum expected throughput by a factor 0.3%, which indicates that the vehicles move throughout the simulation without much congestion.

Experiment 3: Sanction severity

The third experiment illustrates that agents are able to reason about norms. Experiment 1 has shown that agents are norm-aware. However, TrafficMAS agents also have the capabilities to not comply with certain obligations if these are not of significant importance to them. In this experiment the leftmost lane is an emergency lane, reserved for certain traffic in order to help with accidents and other emergencies. Therefore regular agents will get sanctioned if caught driving on this lane. Since this lane remains mostly empty, this is a viable option for agents who greatly value a faster arrival time and are in a financial position which makes them willing to take a fine. We expect that the more affluent agents will choose to take a sanction and win some time, resulting in distinct behavior between the two groups of agents.

Table 7-5: results for experiment 3

	Poor Agents	Affluent Agents
Main road Spawnrate	20%	20%
Ramp Spawnrate	15%	15%
Throughput	20,48	20,88
Average Speed	12,95	14,42
Average Gap	48,37	69,29
Sanctions	0	133,12

This experiment is set up in the same way as experiment 2, with the exception that the leftmost lane is reserved for emergencies and the spawn rates are lowered because of this. In the *Poor Agents* scenario, the input stream consists of agents who are hasty, but in a substandard financial position. The *Affluent Agents* scenario spawns agents who care about being sanctioned, but are willing to take a fine if by doing so they can arrive closer to their goal arrival time with a significant amount of time.

With this experiment, the difference in throughput, average speed and gap is much smaller, and not significant enough to lead to any conclusions about improvement. However, a significant distinction in the number of sanctions can be seen. This indicates a difference in behavior between the groups of agents. On average about 133 affluent agents decide to drive on the emergency lane in an hour of simulation. This shows a clear difference in behavior from the poor agents, who never decided to change lanes, since no sanctions are given to them throughout the 100 runs.

7.7 Contributions and comparison to other work

The contributions of our work are as follows. First of all, we have created a lightweight framework for autonomous norm-aware vehicles and norm-based traffic control system on top of SUMO. This framework is easy to extend with different types of driver profiles. It also allows for the easy usage of a different simulation package. Secondly, with our framework, we have created the possibility to conduct traffic experiments and measuring the impact of a norm-based traffic control system. Finally, we have improved the TraaS performance, yielding a performance increase of up to four times over the original TraaS library in certain scenarios.

Our approach has some similarity with Baines et al [3] since they employ autonomous agents, and also use norms. However, Baines et al concentrate on agents' internal architecture and the communication between agents. Our driver model is deliberately kept simple in order to focus on the interaction between traffic control systems on the one hand, and between agents and traffic control systems on the other hand. Finally, our framework supports decentralized traffic control system while Baines et al. focus on a single, all knowing, institution.

7.8 Future Work

Our framework could be extended in a number of ways. First of all, one could add support for contrary to duty norms. Contrary to duty norms consist of a hierarchy of norms. An agent should comply to all hierarchies, but if it doesn't comply to the first norm level (and thus incurring a sanction), it should at least comply to the second norm level, or get an even higher sanction. An example is the norm "You shouldn't break the speed limit, but if you do, you should drive on the leftmost lane."

Another extension would be to include communicating decentralized traffic control, as described in [8] and [9]. In our current framework, multiple control systems are possible (though not shown in this paper). However, they cannot yet communicate information about vehicles and sanctions with each other. If this infrastructure is created, more complex norms and scenario's can be implemented.

7.9 Conclusion

Our goal was to create a traffic simulation Multi-Agent System where agents should generally follow traffic regulations, yet are able to ignore these regulations in certain circumstances without implementing hard constraints on the agents themselves. We used norm based control systems, since they deal well with these kinds of situations.

Our work is an extension to SUMO. It features a norm based traffic control system which monitors and possibly sanctions the vehicles, as well as deliberative pro-active drivers which make autonomous decisions according to their goal and received sanctions. The extension features i) driver profiles which model different types of behavior, ii) traffic control systems and norms to control these agents and iii) an easy way to add new driver profiles, control systems and norms. This plug and play extension to SUMO can serve as a testing suite for all experiments concerning normative systems.

We showed in our experiments that the performance of normative systems is better than the default behavior in a ramp-merge scenario. Furthermore, we showed that complex norms allow for more fine grained steering of behavior in complicated scenarios. Finally, we

illustrated the autonomy of agents, by demonstrating a different in behavior between poor and affluent agents.

7.10 References

- [1] Alechina, N., Dastani, M., Logan, B.: Programming Norm-Aware Agents. In proc. of the 11th int. conf. on autonomous agents and multi-agent systems, vol. 2, 1057-1064 (2012)
- [2] Alechina, N., Dastani, M., Logan, B.: Reasoning About Normative Update. In proc. of the 23rd int. joint conf. on A.I., 20-26, AAAI Press (2013)
- [3] Baines, V., Padget, J.: A Situational Awareness Approach to Intelligent Vehicle Agents. In SUMO2014 – Modeling Mobility with Open Data, 1-17 (2014)
- [4] Boella, G., Van Der Torre, L.: Introduction to Normative Multi-Agent Systems. Computational and Mathematical Organization Theory, vol.12, 71-79 (2006)
- [5] Hübner, J., Boissier, O., Bordini, R.: A Normative Programming Language for Multi-Agent Organisations. Annals of Mathematics and A.I., vol 62, 27-53 (2011)
- [6] Krauss, S., Wagner, P., Gawron, C.: Metastable States in a Microscopic Model of Traffic Flow. Physical Review E, vol. 55, 5597-5602 (1997)
- [7] Meneguzzi, F., Vasconcelos, W., Oren, N., Luck, M: Nu-BDI: Norm-aware BDI Agents. In proc. Of the 10th Europ. Workshop on Multi-Agent Systems (2012)
- [8] Testerink, B., Dastani, M., Meyer, J.-J.: Norms in Distributed Organizations. In AAMAS'13: Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (2013)
- [9] Testerink, B., Dastani, M., Meyer, J.-J.: Norm Monitoring Through Observation Sharing. In A. Herzig, E. Lorini (eds), Proc. Of the Europ. Conf. on Social Intel., 291-304 (2014)
- [10] Tinnemeier, N., Dastani, M., Meyer, J.-J., van der Torre, L.: Programming Normative Artifacts with Declarative Obligations and Prohibitions. In IEEE/WIC/ACM Int. Joint. Conf. on Web Intel. and Intel. Agent Technologies, vol. 2, 145-152 (2009).
- [11] Baskar, L. D., De Schutter, B., Hellendoorn, J., & Papp, Z.: Traffic control and intelligent vehicle highway systems: a survey. IET Intelligent Transport Systems, 5(1), 38-52 (2011).
- [12] Wang, Ziyuan, Lars Kulik, and Kotagiri Ramamohanarao. "Proactive traffic merging strategies for sensor-enabled cars." Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks. ACM, 2007
- [13] Baines, Vincent, and Julian Padget. "On the benefit of collective norms for autonomous vehicles." Proceedings of 8th International Workshop on Agents in Traffic and Transportation. 2014.
- [14] <http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/tesla-working-towards-90-autonomous-car-within-three-years>
- [15] J. Markoff. Google cars drive themselves, in traffic. <http://www.nytimes.com/2010/10/10/science/10google.html>, October 2010.
- [16] Abdelkader, G.: "Requirements for achieving software agents autonomy and defining their responsibility." Proc. Autonomy Workshop at AAMAS 2003. Vol. 236. 2003.

- [17] Dastani, Mehdi, Davide Grossi, John-Jules Ch Meyer, and Nick Tinnemeier. "Normative multi-agent programs and their logics." In *Knowledge Representation for Agents and Multi-Agent Systems*, pp. 16-31. Springer Berlin Heidelberg, 2009.

7.11 Appendix A: Merge Norm Scheme

For the merge norm scheme we use the same pseudo code structure (Table 7-2) as for the stay-on-lane norm scheme. As in the other norm scheme we begin with the instantiation of the norm (lines 0-8). Initially we read sensors 1 and 3 and merge the readings according to [12] (line 0). The result is an ordered list of agents, which, if they continue as they are, will arrive at the merge point in the same order. We maintain a global variable t_{free} that indicates the next moment in time that the merge point is free. With *optimalVelocity* we calculate the optimal speed for an agent s.t. it will arrive at t_{free} on the merge point plus some safe margin, or later if the agent cannot make it in time physically (line 2). If the agent is at the right lane of the main road and the optimal velocity is below a predefined threshold, then it is obliged to move to the left lane (line 4), otherwise it is obliged to adapt its velocity to the optimal velocity and pass the merge point on the right lane (line 6). An agent is sanctioned if it is not passing the merge point on the correct lane (lines 12-13, and 19-20). Otherwise an agent can also be sanctioned if it did not achieve its predetermined velocity (lines 23-24).

Table 7-6. Pseudo code for the merge norm scheme.

```

0:   $L \leftarrow \text{merge}(\text{read}(s_1), \text{read}(s_3))$ 
1:  for each  $\text{agent} \in L$  do {
2:       $s \leftarrow \text{optimalVelocity}(\text{agent}, t_{free})$ 
3:      if  $\text{agent.lane} = l_{right} \ \& \ s < v_{threshold}$  then {
4:           $NI \leftarrow NI \cup \{(\text{agent}, (l_{left}, v_{MAX}), fine_{high})\}$ 
5:      } else {
6:           $NI \leftarrow NI \cup \{(\text{agent}, (l_{right}, s), fine_{high})\}$ 
7:      }
8:  }
9:   $L \leftarrow \text{read}(s_4)$ 
10: for each  $\text{agent} \in L$  do {
11:      $NI \leftarrow NI \setminus \{(\text{agent}, (l_{left}, v_{MAX}), fine_{high})\}$ 
12:     if  $(\text{agent}, (l_{right}, s), fine_{high}) \in NI$  then {
13:          $\text{sanction}(\text{agent}, fine_{low})$ 
14:          $NI \leftarrow NI \setminus \{(\text{agent}, (l_{right}, s), fine_{high})\}$ 
15:     }
16: }
17:  $L \leftarrow \text{read}(s_5)$ 
18: for each  $\text{agent} \in L$  do {
19:     if  $(\text{agent}, (l_{left}, v_{MAX}), fine_{high}) \in NI$  then {
20:          $\text{sanction}(\text{agent}, fine_{high})$ 
21:          $NI \leftarrow NI \setminus \{(\text{agent}, (l_{left}, v_{MAX}), fine_{high})\}$ 
22:     } else if  $(\text{agent}, (l_{right}, s), fine_{high}) \in NI$  then {
23:         if  $s \neq \text{agent.v}$  then {
24:              $\text{sanction}(\text{agent}, fine_{high})$ 
25:         }
26:          $NI \leftarrow NI \setminus \{(\text{agent}, (l_{right}, s), fine_{high})\}$ 
27:     }
28: }
```