A Directable Vehicle Behavior Model for Virtual Driving Environments

James Cremer, Joseph Kearney, Peter Willemsen
Computer Science Department
University of Iowa, Iowa City, IA 52242, U.S.A.
{cremer,kearney,willemsn}@cs.uiowa.edu

Abstract

In this paper we present a model for autonomous driving behavior useful for creating ambient traffic as well as experiment specific scenarios for driving simulation. This model follows roadways, obeying the rules of the road. It reacts to nearby vehicles and traffic control devices. The model supports a range of behaviors including passing, lane changes, and safe navigation through intersections. The model is parametrized by traits that influence the personality of the driver such as aggressiveness, attentiveness, impatience, and law-abidingness. In addition, the model responds to behavior-oriented directives that can be used to coordinate groups of vehicles for the creation of complex scenarios. In this paper, we present the details of the behavior model and an example of its use in creating a scenario.

1 Introduction

Interactive driving simulation requires microscopic simulation of driving behavior for synthetic vehicles that is consistent and believable. Vehicle motions must be smooth and continuous. Cars cannot discretely jump from lane to lane as they typically do in macroscopic driving simulators. Within this framework, it is important to control the global properties of traffic such as density and flow. These properties must naturally fall out of the behaviors of individuals. In addition, it is important to create specific situations and circumstances that require coordinated actions of groups of vehicles to occur within a backdrop of ambient traffic.

In this paper, we demonstrate how a single, computational framework can be used to generate interesting scenarios from ambient traffic composed of microscopically simulated autonomous vehicles. We present a detailed driving behavior model and an example of its use in creating an interesting scenario. In Section 2 we outline HCSM, the modeling framework used to define behaviors and scenarios. Section 3 describes the HCSM model for driving behavior, as well as a directable traffic light model. Use of these models is demonstrated in an intersection hazard scenario in Section 4.

1.1 Related work

The development of appropriate driving models and scenario creation tools for virtual driving environments draws on aspects of both macroscopic and microscopic traffic simulation. TRAF-NETSIM [11] is a well-known program for macroscopic simulation of urban traffic. Work in the SmartPATH project[5, 4] encompasses both macroscopic traffic management and microscopic-level vehicle modeling, simulation, and control in order to support evaluation of Intelligent Vehicle Highway Systems (IVHS). Examples of microscopic driving models suitable for real-time virtual environment applications can be found in [12, 3, 13].

2 The Hierarchical Concurrent State Machine (HCSM) Framework

In this section we briefly outline the HCSM programming framework[1]. HCSM is based on communicating, hierarchical state machines that grew out of our experiences programming the control of high-degree-of-freedom mechanisms in multibody dynamics simulations [6, 8] and working with vehicle models and scenarios in the Iowa Driving Simulator[9]. We built HCSM around an extended state machine model that facilitates creation of scenarios and complex autonomous agents by providing abstraction, communication, and arbitration mechanisms.

State machines of various forms have long been used to model reactive behavior. Harel[7] provides an excellent description of the weaknesses of single-level finite state machines for programming complex reactive systems, and introduces the Statecharts formalism to address them. The hierarchical concurrent state machines of HCSM are similar to Statecharts in several ways. An HCSM state machine can be defined recursively; it is either a *leaf* machine or contains a number of child HCSM state machines. In a *sequential* HCSM state machine, only one child HCSM is active at a time.[1] The active child HCSM can change based on the firing of *transitions*. In a *concurrent* HCSM state machine, all children are active at once and there are no transitions. Concurrency allows behaviors to be decomposed into a set of possibly, but not necessarily, orthogonal components. When combined with hierarchical abstraction, this significantly reduces the complexity of programming behaviors.

HCSM state machines provide a different communication mechanism than Statecharts. Conceptually, a state machine has a control panel consisting of "buttons" and "dials" that provide a communication interface between the state machine and the world outside it. The behavior of the state machine may be influenced by sending messages to the machine, "pushing" buttons or "setting" dials to achieve some desired effect. Buttons and dials address the critical need for means to create agents that are not purely reactive, but instead are partly reactive and also controllable or directable.

HCSM state machines also differ from Statecharts by explicitly incorporating a notion of the *activity* performed by the state machine. Whenever an HCSM machine is executed, it outputs a value (or set of values) determined by the *activity function* associated with the machine. Specifically, an activity function computes an HCSM's output as a function of the values of (1) local variables, (2) input parameters, (3) messages received by buttons and dials, and (4) values output by child HCSMs. The values output by top-level HCSM machines are used as control inputs for entities modeled in the simulation environment. Activity functions provide a means for arbitrating between or *resolving* the possibly conflicting values output by concurrent child state machines competing to control the same resources. Activity functions are also useful in sequential state machines, since they provide a way to modify, filter, or otherwise change values passed up from lower level behaviors.

As demonstrated below, HCSM provides a uniform framework both for modeling the basic entity behaviors (e.g. driving) as well as for coordinating and directing the actions of multiple agents in scenarios. We call HCSM state machines used for the latter purpose *directors*. See [1] for a detailed definition and description of HCSM.

[1]For convenience, we often refer to the child state machines of a sequential HCSM as the *states* of the machine.

3 Behavior Models

For basic driving experiments we need to model the autonomous behavior of vehicles and traffic lights. In this paper, we focus on a driving environment that includes one- and two-lane city streets and one lane rural highways. We assume that vehicles can access a database to gain information about nearby vehicles, road paths, roadway logic, and the state of upcoming traffic lights. In our system, a separate software component provides this information[2, 10]. The outputs of the HCSM behavior models are used as control inputs to physical models of entities. For a traffic light, this is the color of the light at the current time. Vehicle behavior models output values to control acceleration and heading. The output values are passed to a kinematic or dynamic vehicle model. At present, we use a kinematic vehicle model that computes a new position, heading, and velocity based on the control values and the current position, heading, and velocity.

We first describe a directable traffic light model and follow with a vehicle behavior model. The traffic light model serves as a simple example in which to introduce the HCSM methodology and notation.

3.1 Traffic Light Behavior Model

Traffic lights are modeled with a simple sequential state machine that cycles through three states for the red, green, and yellow signals. Internal (local) variables specify the intervals spent in each phase during normal operation.

Experiments frequently call for the pattern of light changes to be synchronized with the approach of the subject's vehicle. For example, we may want a light to turn yellow a short time before the subject reaches an intersection. It is important to give the subject the impression that the traffic light is operating normally – the timing of the phase change should appear to be happenstance.

The traffic light HCSM supports three kinds of input that control its behavior. Two of these have an immediate effect on the conditions that regulate state transitions. The third causes a change in the timing sequence for a temporary period to meet a target event at a future time. The immediate behavior of the light can be influenced by messages to either change to the next phase or stay in the current phase. These

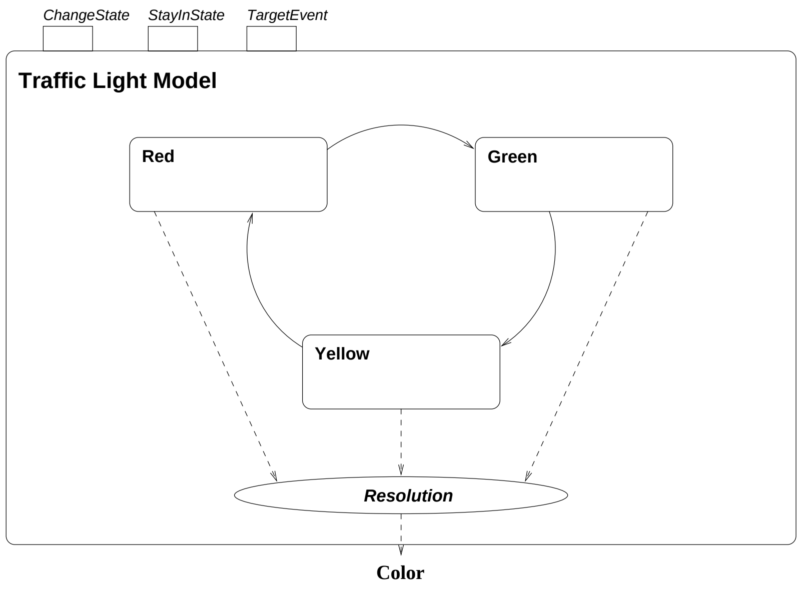

Figure 1: HCSM traffic light model

mechanisms provide a simple, direct means to change the phase of the light. However, for many experiments it is important to create enabling conditions that require greater anticipation of the upcoming event. The sequencing pattern can be modified by specifying a future time at which a specific transition is to take place. The state machine will uniformly compress or expand intervals between the current time and the given time so that the correct transition takes place at the given time. The normal progression of states is maintained and the smallest possible change is made in the normal cycle that will place the transition at the desired time.

It is usually not possible to anticipate the exact time of a future event in an interactive simulation. Subjects vary their speeds in unpredictable ways and encounter different ambient traffic. However, we can progressively refine the timing as the subject nears the intersection. By continually feeding the current best estimate of the event time, the traffic light can make fine adjustments to the scheduled sequence of changes and accurately meet the required deadline.

3.2 Vehicle Behavior Model

Figures 2 and 3 present a directable HCSM state machine modeling driving behavior for navigating city streets and rural highways. It reacts to the movement of surrounding traffic and upcoming traffic lights. The structure of the driving state machine is shown in Figure 2. The flow of data through the machine and the resolution of competing outputs from concurrent child state machines is shown in Figure 3.

The top level state machine has four dials that can be set by external directors to influence the driving behavior of the vehicle. The *Speedfactor* dial influences vehicle's proclivity to follow the posted speed limit. Positive settings tend to increase normal driving speeds and negative settings tend to decrease normal driving speeds. The *Changelane* and *Turn* dials influence the vehicle's disposition to change lanes or turn at the next intersection. The *IgnoreLight* dial instructs the state machine to disregard traffic lights. In the example below, we demonstrate how these dials can be used to coordinate the motions of a group of vehicles to create conditions for a scenario.

Driving behavior is decomposed into sub-behaviors responsible for cruising on an open road, vehicle following, lane tracking, turning at intersections, and highway passing. The CRUISE and FOLLOW state machines control the speed of the vehicle along a continuous stretch of road. The TRACKLANE state machine is responsible for steering the vehicle so that it tracks the center line of a lane. It also encapsulates decisions to change lanes and steers the vehicle from lane to lane. The TURN and PASS state machines are concerned with negotiating turns at intersections and managing highway passing maneuvers, respectively. In the remainder of this section we describe the inner workings of the child state machines that contribute to driving behavior and explain how their output is integrated by the resolver (*i.e.* activity function) to produce safe

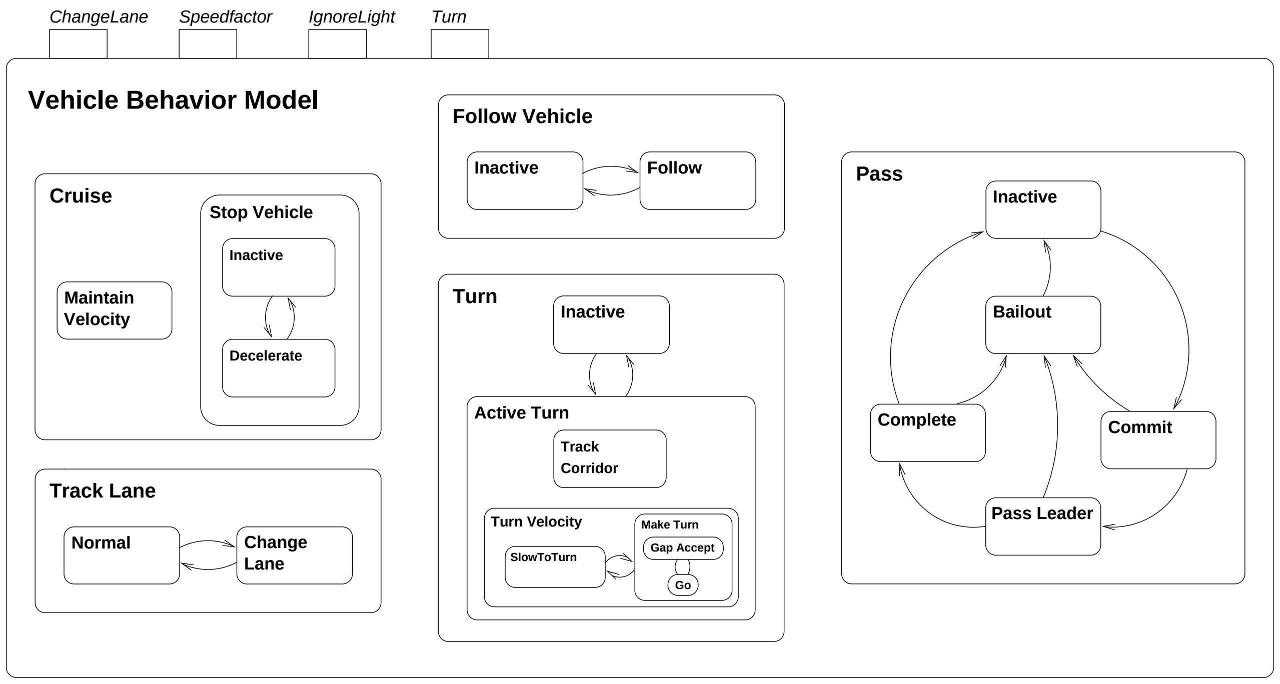

Figure 2: Vehicle behavior structure

driving behavior.

Sub-behavior models

The CRUISE state machine attends to posted speed limits and upcoming traffic lights. When no traffic light is visible, it computes an acceleration proportional to the difference between the current speed and its desired speed. The desired speed is based on the posted limit and the *SpeedFactor* input which determines the vehicle's tendency to comply with speed regulations. The child machine STOPVEHICLE is in the INACTIVE state and produces a null output value whenever the vehicle is distant from an intersection. As the vehicle approaches an intersection the DECELERATE state is activated to bring the vehicle to a stop. The resolver places higher priority on the output of the STOPVEHICLE child machine whenever it produces a non-null value. By itself, the CRUISE state machine could drive along one lane roads with no other traffic present.

The FOLLOWVEHICLE state machine is responsible for maintaining a safe separation between the vehicle and the car traveling ahead it in the same lane. It is activated when the headway is less than minimum acceptable separation based on the time to close the gap at the current speed. When active, the state machine uses a PD controller to maintain a speed-dependent separation distance.

The role of the TRACKLANE state machine is to compute steering control values that will track a lane of the road. This state machine also steers the vehicle between adjacent lanes that have same direction of traffic flow. In state NORMAL, TRACKLANE queries the road database to determine the path of the road on which the vehicle is driving. It determines a target position on the lane centerline to which the vehicle should head and then calculates a change in the steering angle necessary to direct the vehicle to the target position.

On multi-lane roads, the vehicle may decide to change lanes. The intent to change lanes may be due to the presence of a slow moving vehicle interfering with travel or may be stimulated by input from the parent state machine. Before a lane change can be initiated there must be an opening in an adjacent lane into which to move. Given the intention and an opening, the state machine will make a transition to the CHANGELANE state machine. This state produces changes in steering angle that will guide the vehicle into the adjacent lane. Once the vehicle has completed the lane change, the TRACKLANE state machine sends the identifier of the new lane as output. The parent state machine uses this information to update a local variable that records the lane logically occupied and tracked by the vehicle.

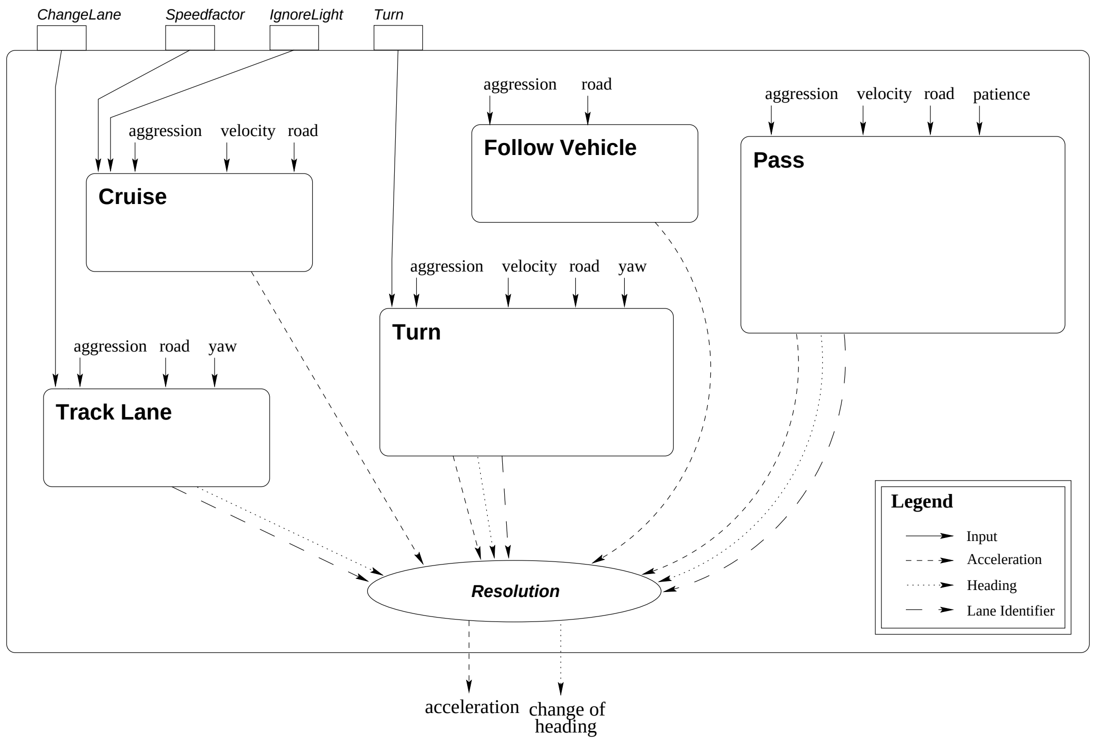

Figure 3: Vehicle behavior dataflow

The TURN state machine is responsible for negotiating turns at intersections. This machine waits in a dormant state, producing a null output value, until the vehicle nears an intersection. At this time, a decision is made to either turn at the intersection or to continue on the same road through the intersection. The turn decision is based on the road density, road type, and the internal goals of the state machine. The decision is also influenced by input from the parent state machine.

If a decision is made to turn, then a transition is taken to the ACTIVETURN state. The two child machines, TRACKCORRIDOR and TURNVELOCITY, control steering and acceleration, respectively, as the vehicle approaches and drives through the intersection. The TRACKCORRIDOR machine outputs changes in steering angle to track turning corridors defined in the database. The SLOWTOTURN state machine monitors the speed of the vehicle as it approaches the intersection and produces acceleration values to appropriately prepare the vehicle for turning. The TURNVELOCITY machine is responsible for managing the speed of the vehicle through the intersection. On turns that require the vehicle to traverse lanes of opposing traffic (e.g. left turns on two-way streets), the TURNVELOCITY state machine must find a gap in the oncoming traffic through which the vehicle can safely drive. When no gap exists, the GAPACCEPT state causes the vehicle to enter into the intersection and pause until a gap opens. When a gap is found, the GO state produces accelerations to drive the vehicle through the intersection.

The PASS state machine executes passing maneuvers on two lane, rural highways. As with lane changes, passing requires both an intention to pass and the correct enabling circumstances. The motivation to pass derives from a combination of an urge to drive faster than the vehicle being followed and personality characteristics such as aggression and patience. To enable passing, the road must be marked to permit passing and there must be a gap in the oncoming traffic sufficiently large to perform the maneuver. Given both the intention and an opportunity to pass, the state machine enters the COMMIT state. This state produces both an acceleration and a change in steering angle to drive the vehicle into the passing lane. Once the vehicle is situated in the passing lane, control passes to the PASSLEADER state that propels the vehicle ahead of the vehicle to be passed. When the vehicle is ahead of the vehicle to be passed, COMPLETE becomes the active state. This states controls reentry into the normal lane.

After the vehicle initiates the pass it continues to monitor oncoming traffic while in the COMMIT, PASSLEADER, and COMPLETE states. If an oncoming vehicle poses a serious threat of colliding with the vehicle, a transition is taken to the BAILOUT state. The BAILOUT state chooses the quickest escape route out of the passing lane. This may involve a hurried completion of the pass or a radical deceleration in an attempt to abort the pass and slide behind the vehicle that was to be passed.

Resolution of competing behaviors

The resolution function must arbitrate among the five child machines that compete for control of the vehicle. Because child state machines generate control values to satisfy specific goals and circumstances (e.g. don't collide with the vehicle ahead or stop for red lights), it is generally not sensible to combine output values in an attempt to partially satisfy the concerns of several child machines. The combined value typically leaves one or many goals unmet and can lead to disastrous outcomes. For example, if the resolution function averaged the accelerations computed by the CRUISE and FOLLOW state machines it would cause the vehicle to ram slow moving vehicles it encountered on high-speed roadways. The more effective strategy is to assign control to a single child state machine on the basis of the values produced or by following predetermined priorities.

In most circumstances, the demands of safe driving place upper limits on the speed at which we travel – our speed should be less than the posted limit, we slow for sharp turns, we slow to stop, and we slow for pokey drivers. It is rarely the case that a circumstance justifies suspension of these limits to accomplish a task or react to a critical situation. Most of the time, the cautious strategy of driving at the maximum speed that satisfies all the constraints imposed by the environment will move us expeditiously and safely along the road. Our resolution function accomplishes this by using a "most conservative" rule that chooses the minimum acceleration output value.

An exception to the "most conservative" rule is made for passing. While passing, the passing vehicle may drive faster than the posted limit in order to overtake the vehicle to be passed. In order to coordinate the stages of the passing maneuver, the passing sub-state machine is assigned complete control of the vehicle for the duration of the maneuver. By assuming control, the passing sub-state machine must accept responsibility for all aspects of driving. A drawback of this approach is that it can lead to duplication of behaviors modeled in other state machines. For example, if vehicles are allowed to pass in gangs, then the passing state machine must include its own treatment of following behavior to avoid colliding with other passing vehicles. However, we found the benefit of coherence in the stages of passing outweighed possible redundancies.

In contrast to speed control, which involves simultaneous attention to many factors, steering control typically involves mutually exclusive tactics. The driver chooses to either track the current lane, change lanes, turn at an intersection, or pass. Each tactic prescribes a separate action. Some of these tactics can only be used in highly restricted situations – the vehicle can only turn at intersections and should not pass near intersections. We leave the context dependent determination of the appropriateness of a tactic to the internal decision making apparatus of the state machines. The resolution function imposes an ordering on the state machines that produce steering output values: PASS, TURN, TRACKLANE. Whenever two or more state machines produce non-null values, priority is given to the machine that appears first in the order. Thus, PASS supersedes TRACKLANE when PASS is producing steering output.

4 An Example Scenario

In this section we demonstrate how to build a complex scenario using this vehicle model as the central component. Directors are used to detect events and modify the actions of vehicles and a traffic light to create a consistent and cohesive experience.

Our example scenario involves a subject's response to a vehicle driving through a red light into the path of the oncoming subject's vehicle. It is not possible to choose the vehicle to perform the violation off-line because of possible interferences in the traffic flow. Because human subjects drive at different rates they will arrive at the intersection at different times. Thus, we cannot guarantee that a particular car will be properly positioned, waiting at the intersection for the red light on the crossing road, when the subject arrives. Instead, a director is used to conscript an appropriate scenario vehicle to run the light as the subject nears the intersection. The scenario requires coordination of the ambient traffic to create consistent circumstances for the event.

Figure 4 illustrates how three concurrent directors can be used to direct the traffic surrounding the driver, synchronize phases of the traffic light to the approach of the subject's vehicle, and select a vehicle on the

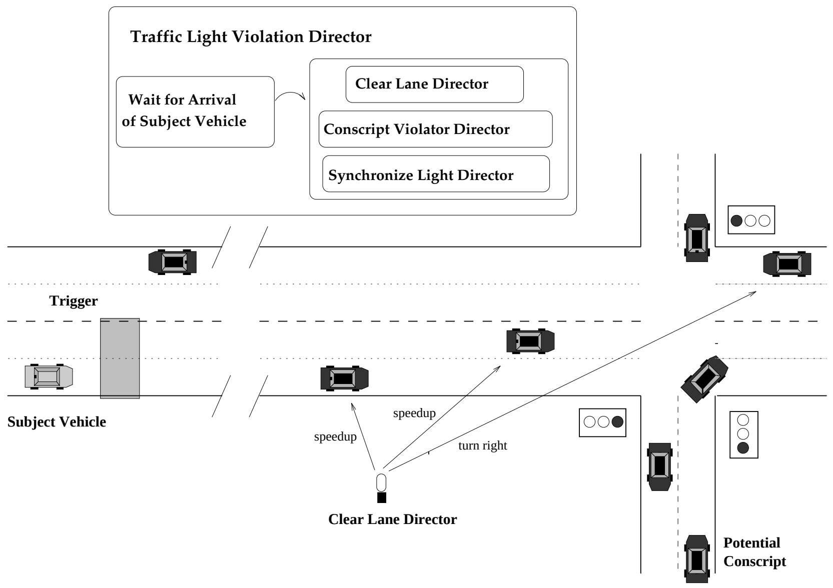
Traffic Light Violation Director
Wait for Arrival of Subject Vehicle
Clear Lane Director
Conscript Violator Director
Synchronize Light Director
Trigger
Subject Vehicle
speedup
speedup
turn right
Clear Lane Director
Potential Conscript

Figure 4: Multiple directors can be used to orchestrate complex situations involving large numbers of objects.

crossing road to perform the violation. All of the directors are activated by a trigger in the road that senses the approach of the subject vehicle.

To create a clear path in front of the subject vehicle that will allow the violator to enter the intersection, the **Clear Lane Director** sets the *Speedfactor* dials on vehicles ahead the subject's vehicle, instructing them to increase their speed. To give the subject room to maneuver, the *Speedfactor* of vehicles behind the subject vehicle is decreased and oncoming traffic is told to turn right. Thus, a pocket of vacant road is created around the subject vehicle.

The **Synchronize Light Director** is responsible for synchronizing the sequencing of the traffic light to the approach of the subject vehicle. In this scenario, we chose to have the light be in a green state when the subject reaches the intersection. This is accomplished by estimating the subject's arrival time at the intersection and sending this information to the traffic light model via the *TargetEvent* message.

Lastly, the **Conscripted Vehicle Director** monitors the distance between the subject vehicle and the upcoming intersection. As the subject approaches the intersection, a vehicle in the crossing road is selected to perform the violation. At a moment near the time when the subject vehicle reaches the intersection, the violator is sent the *NoStop* message forcing it to enter the intersection.

5 Conclusion

In this paper, we presented directable models for driving and traffic light behavior suitable for use in real-time virtual driving environments. We demonstrated how the HCSM framework was used both to model basic driving behavior and to define directors used in orchestrating scenarios. This work presents our first attempt to combining autonomy and directability in behavior models. There is much yet to learn about modeling directable driving behavior, and about creating scenarios involving complex sequences of events. We believe that experimentation will form a substantial part of this learning process.

Acknowledgements

This research was supported in part by National Science Foundation grant IRI-9506624.

References

[1] James Cremer, Joseph Kearney, and Yiannis Papelis. HCSM: A framework for behavior and scenario control in virtual environments. *ACM Transactions of Modeling and Computer Simulation*, July 1995.

[2] James F. Cremer, Joseph K. Kearney, Yiannis Papelis, and Richard Romano. The software ar-

chitecture for scenario control in the Iowa Driving Simulator. In *Proc. 4th Computer Generated Forces and Behavioral Representation Conference*, Orlando, FL, May 1994.

[3] S. Donikian and B. Arnaldi. Complexity and concurrency for behavioral animation and simulation. In *Proceedings of the 5th Eurographics Workshop on Animation and Simulation*, pages 101–113, Oslo, September 1994.

[4] F. H. Eskafi and D. Khorramabadi. Modeling the interactive mode of smartpath. In *Proceedings of the 1994 Conference on AI, Simulation, and Planning in High Autonomy Systems*, pages 251–256, Gainesville, December 1994. IEEE Computer Society Press.

[5] Aleks Göllü, Akash Deshpande, Praveen Hingorani, and Pravin Varaiya. Smartdb: An object-oriented simulation framework for intelligent vehicles and highway systems. In *Proceedings of the 1994 Conference on AI, Simulation, and Planning in High Autonomy Systems*, pages 244–250, Gainesville, December 1994. IEEE Computer Society Press.

[6] Stuart A. Hansen. *Conceptual Control Programming for Physical System Simulation.* PhD thesis, Computer Science Department, University of Iowa, May 1993.

[7] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[8] Joseph K. Kearney, Stuart Hansen, and James F. Cremer. Programming mechanical simulations. *The Journal of Visualization and Computer Animation*, 4(2):113–129, April-June 1993.

[9] J. Kuhl, D. Evans, Y. Papelis, R. Romano, and G. Watson. The Iowa Driving Simulator — an immersive research environment. *IEEE Computer*, pages 35–41, July 1995.

[10] Y. Papeli and S. Bahauddin. Logical modeling of roadway environment to support real-time simulation of autonomous traffic. In *Proceedings of the First Workshop on Simulation and Interaction in Virtual Environments*, pages 62–71. The University of Iowa, July 1995.

[11] Ajay K. Rathi and Alberto J. Santiago. Urban network traffic simulation: Traf-netsim program. *Journal of Transportation Engineering*, 116(6):734–743, 1990.

[12] Douglas A. Reece. *Selective Perception for Robot Driving.* PhD thesis, Carnegie Mellon University, May 1992.

[13] Rahul Sukthankar. Situational awareness for driving in traffic. PhD Thesis Proposal. Robotics Institute. School of Computer Science. Carnegie Mellon University, October 1994.