Statechart-based representation of hybrid controllers for vehicle automation

J.-S. Lee and P.-L. Hsu

Abstract: Recently, designs for the automated highway systems (AHS) to improve the safety and efficiency of highways have attracted much attention. However, such large-scale and complex dynamic systems present a number of control issues. Part of the complication comes from the hybrid nature of these systems, which consists of both time-driven and event-driven dynamics. In this paper, an object-oriented representation towards a hybrid controller design for automated vehicles in an AHS is proposed. The traditional discrete statechart of unified modelling language is extended to model the continuous dynamics of hybrid controllers in a hierarchical and natural representation. The proposed modelling approach provides a clear and natural representation of the interaction between the discrete and continuous behaviours.

1 Introduction

Intelligent transportation systems (ITS), formerly called intelligent vehicle-highway systems, aim to improve the efficiency of current transportation systems by applying modern technology. One part of the ITS program, the automated highway system (AHS), promises to reduce traffic congestion and increase the safety, efficiency and capacity of highway systems without building additional highways. It does this by adding intelligence to both the vehicle and the roadside. The AHS control structure is centred on the concept of a platoon, which is a group of tightly spaced vehicles following a leading vehicle at highway speeds. The tight spacing between vehicles must be maintained without negatively impacting passenger safety (and comfort if possible).

The AHS control structure, shown in Fig. 1, is organised in a hierarchy with five layers [1]. In the infrastructure system, the top network layer evaluates the traffic status of the entire highway system, calculates the desired densities and speeds to prevent congestion, and sends the appropriate commands to the link layer. Then, the link layer calculates an optimum platoon size and velocity for each section to maximise throughput. In the vehicle system, the coordination layer coordinates the operation of platoons with their neighbours. It receives the link-layer commands and translates them to specify manoeuvres such as the join, split and lane change [2]. Then, the regulation layer receives the manoeuvre requests and computes the necessary commands to control the throttle, brake and steering actuators within the vehicles. The bottom physical layer is the actual automated vehicle with sensors that provide sampled information for the control algorithm.

Clearly, such a large-scale system poses a very complex control problem. Part of the complication comes from its hybrid nature. For example, the coordination layer is a discrete event system (DES) that supervises the regulation layer to directly control the vehicle, which is a continuous variable system (CVS), and, thus, the coordination and regulation layer controllers within the automated vehicles form a hybrid control system.

During the past several years, a great deal of work has gone into developing continuous controllers that the regulation layer uses to perform the various manoeuvres [3–9]. Also, coordination-layer protocols have been designed and extended to fault-tolerant manoeuvres [2,10,11]. However, there is still a gap between these two areas that prevents their smooth integration into a hybrid system. Generally, modern research on system modelling and control is mainly focused on either the CVS or the DES. The physical dynamic behaviour of the former is typically modelled and analysed by differential or difference equations, while the latter is described using various frameworks, such as finite state automata, Petri nets and max-min algebra [12–15], which can capture logical and sequential behaviour. However, real applications of AHS are hybrid dynamic systems (HDS) that combine both time-driven and event-driven dynamics and the suitable approach for modelling such systems is thus required [16,17].

In this paper, we extend the standard discrete statechart of unified modelling language (UML) [18] to model hybrid systems and proceed to develop a hybrid controller for automated vehicles. The result shows that the proposed modelling approach provides a clear and natural representation of the interaction between the discrete and continuous behaviours.

2 Statechart-based representation

2.1 Statecharts

Statecharts constitute a visual formalism for describing states and transitions in a modular fashion. It extends

IEE Proceedings online no. 20060018

doi:10.1049/ip-its:20060018

Paper first received 20th March 2006 and in final revised form 18th July 2006

Jin-Shyan Lee is with the Information & Communications Research Labs, Industrial Technology Research Institute, Hsinchu, Taiwan

Pau-Lo Hsu is with the Department of Electrical & Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan

E-mail: jinshyan_lee@itri.org.tw

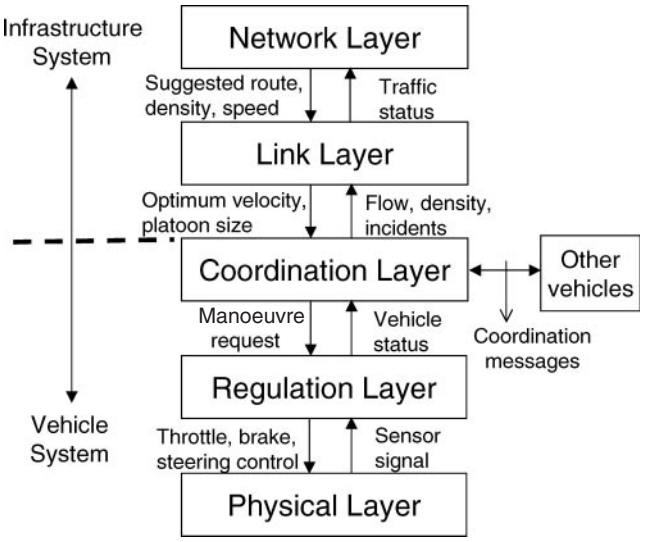

Fig. 1 *The hierarchical control structure of AHS [1]*

the classical formalism of finite-state machines and state transition diagrams by incorporating the notions of hierarchy, orthogonality (concurrency), a broadcast mechanism for communication between concurrent components, composition, aggregation, and refinement of states [19]. Formally, a statechart is defined as

$$\text{Statechart} = (E, S, h, c, \Delta, A),$$

where,

- E is a finite set of basic events.
- S is a finite set of state names.
- $h \in S \rightarrow 2^S$ is the hierarchy function. It defines for each state the set of children of that state.
- $c \in S \rightarrow \{ PRIM, AND, OR\}$ is the class function. It gives for each state its type: primitive state, AND-state or OR-state. Primitive states have no children. AND-states consist of two or more substates graphically separated by dashed lines. They introduce concurrency. OR-states contain a finite state machine with one or more substates.
- $\Delta \in S \rightarrow 2^S$ is the initial state function. It also gives the default substates of a state.
- A is a finite set of arrows.

A state is denoted as a rectangle with rounded corners and a transition is represented as an arrow connecting states. Taking Fig. 2 as an example, we have a statechart with the elements as follows:

- $E = \{Switch\ on, Switch\ off, Start\ Proc\text{-}A, End\ Proc\text{-}A, Start\ Proc\text{-}B, End\ Proc\text{-}B\}$;
- $S = \{OFF, ON, Sa, Sb, Idle, Busy\}$;
- $h(OFF) = \{\emptyset\}$, $h(ON) = \{Sa, Sb\}$, $h(Sa) = h(Sb) = \{Idle, Busy\}$, and $h(Idle) = h(Busy) = \{\emptyset\}$;
- $c(OFF) = \{PRIM\}$, $c(ON) = \{AND\}$, $c(Sa) = c(Sb) = \{OR\}$, and $c(Idle) = c(Busy) = \{PRIM\}$;
- $\Delta = \{OFF\}$ and $\Delta(Sa) = \Delta(Sb) = \{Idle\}$.

Note that from a global point of view, a statechart can be considered as a macro state, which may not be defined in S. Thus, $\Delta = \{OFF\}$ indicates the initial condition of the statechart is OFF, while $\Delta(Sa) = \Delta(Sb) = \{Idle\}$ means the initial conditions of state Sa and Sb are both $Idle$.

2.2 Extension to HDS

The UML is a recent product generated by the aggregation of previous object-oriented methodologies for software engineers [18]. The statechart of the UML is the primary means for capturing complex dynamic behaviours. It is inherently a language suitable for DES and handles the discrete behavioural representation for real systems. A traditional statechart represents a state machine and the sequences of states together with its responses and actions. Since it is basically developed for DES modelling, in order to capture the continuous dynamics, its extension for HDS modelling is proposed as shown below.

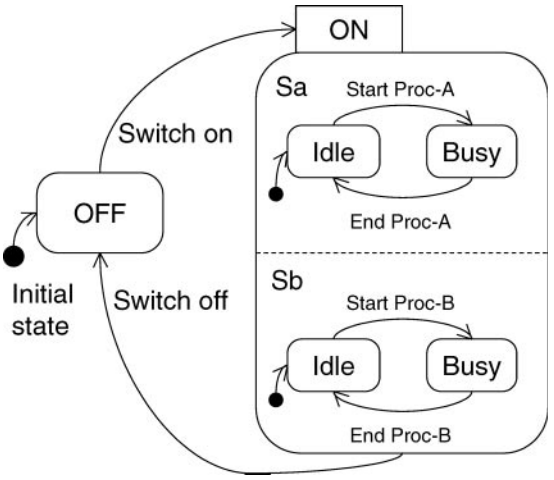

Fig. 2 *A simple statechart*

$$\text{Extended Statechart} = (E, S, h, c, \Delta, A, f),$$

where, $f \in S \rightarrow \{x(t), x(k), X(L), X(Z)\}$.

The physical continuous dynamics associated with a state remains as the physical equations, i.e. differential or difference equations, denoted as $x(t)$ or $x(k)$. Also, instead of equations, transfer functions, i.e. Laplace or Z-transfer functions, can be also used, denoted as $X(L)$ or $X(Z)$. When a state changes as a result of some discrete events, the continuous behaviour may also change. In turn, a condition specified on continuously changing variables may also trigger an event for state transitions.

To distinguish clearly between discrete and continuous states, the continuous state is represented as a double rectangle with rounded corners. To identify clearly an event for transmitting or receiving continuous variables, its corresponding transition is described as an arrow with thick line, which could be recognised as a message pipe. A continuous variable state may be associated with either the physical equations, transfer functions or a message pipe, as shown in Fig. 3.

2.3 Modelling procedure

The statechart modelling procedure for the HDS consists of the following steps:

Step 1) Based on the specification, divide the system into several modules.
Step 2) According to these modules, sketch a high-level statechart.
Step 3) For each module, define the discrete states (S), events (E) and then connect the states with the arcs (A) in a logical order.
Step 4) For each state with continuous dynamics, define the associated physical equations (f), transfer functions or other convenient means.
Step 5) Based on Δ, set the initial conditions.

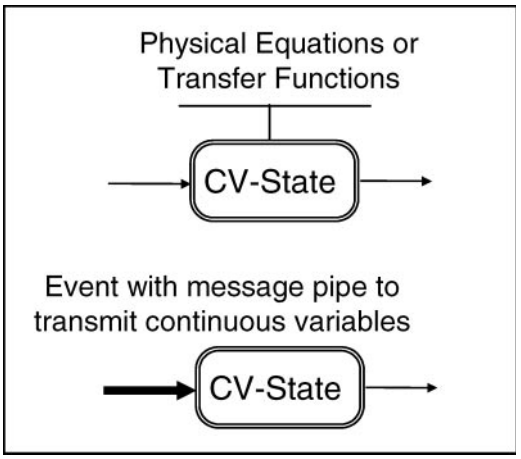

Fig. 3 *The continuous variable state in the extended statechart*

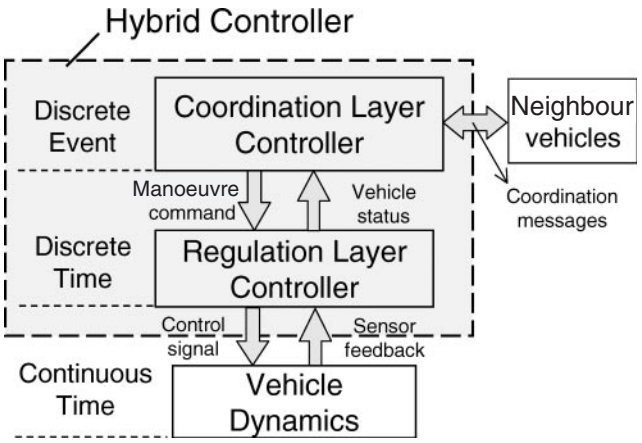

Fig. 4 *The hybrid controller for automated vehicles in an AHS*

Basically, this is a top–down approach for modelling a system in a hierarchical way, i.e. from high-level states to low-level ones, and from discrete behaviours to continuous dynamics. In addition, for large-scale systems, Step 1 provides a modular way to handle effectively the complexity. Also, the refinement can be applied to modify the models.

3 Hybrid controllers in automated vehicles

The block diagram of the vehicle controller for automated vehicles is shown in Fig. 4. It is a typical hybrid control system, where a discrete-event coordination controller supervises and directs a discrete-time regulation controller so as to control a continuous-time system, in this case the vehicle dynamics. Thus, the coordination and regulation layer controllers within the automated vehicles form a hybrid control system.

3.1 Coordination layer design

The task of the coordination layer is to systematically organise the traffic into platoons. It is assumed that the coordination layer knows the target speed, platoon size and path assigned by the link layer, and the vehicles are equipped with communication devices to exchange information such as their current vehicle positions in the highway section. Based on this information, the coordination layer decides which manoeuvre to perform and then exchanges the messages with the relevant neighbouring platoon leaders to coordinate their movement so that the vehicle can safely carry out the manoeuvre. For safety and simplicity, only one manoeuvre is allowed to be carried out at a time, and only the leader or free agent can initiate a manoeuvre. As shown in Fig. 5, the three typical manoeuvres are the join, split and lane change [2]. Basically, an automated vehicle is either a leader of a platoon, a follower in the platoon or a free agent (one-car platoon) at any moment of time.

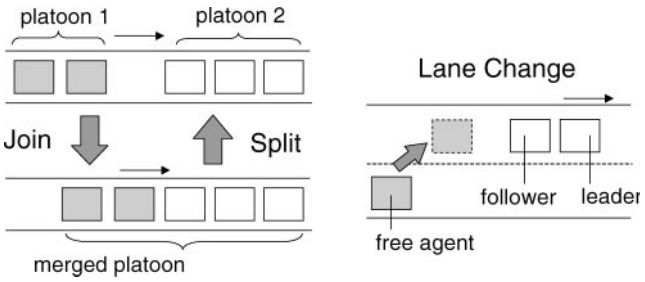

Fig. 5 *The join, split and lane change manoeuvres*

1) Join: The join manoeuvre is used to merge two platoons in the same lane to form a single platoon. The following platoon requests permission from the leading platoon to join. If the leading platoon is not engaged in another manoeuvre and the resulting platoon size does not exceed the upper limit set by the link layer, the permission will be granted. Then, the coordination layer of the following platoon will direct its regulation layer to accelerate and catch up with the leading platoon.

2) Split: The split manoeuvre performs the opposite operation as the join manoeuvre. It breaks one platoon into two smaller platoons.

3) Lane change: The lane change manoeuvre is used to move vehicles from one lane to another. The regulation layer of the vehicle decelerates to adjust its longitudinal position and then performs a lateral movement to change lanes. For safety and simplicity, it is assumed that only free agents are allowed to change lanes. Thus, if the leading or following vehicle wishes to make a lane change, it should first request and perform the split manoeuvre to become a free agent.

3.2 Regulation layer design

The design of the above coordination layer supposes that the regulation layer has the capacity to implement six types of feedback control laws in order to carry out the above manoeuvres. The feedback control laws use sensory information to produce throttle, brake and steering inputs for the vehicle actuators. They can be grouped into the longitudinal laws, for movement along a single lane [3–7], and the lateral laws for movement across lanes and lane tracking [8,9]. The vehicle always operates under one longitudinal law and one lateral law. It should be noted that these are general control laws, each of which may include a variety of different laws involved in its implementation. The operations of the laws are described as follows:

1) Leader law: When a vehicle is a leader or a free agent in an AHS, it must maintain a desired spacing between platoons and try to track the optimum velocity assigned by the link layer as closely as possible.

2) Follower law: This law makes a following vehicle keep a required spacing from the preceding vehicle in its platoon. Inside a platoon all the vehicles follow the leader with a small intra-platoon separation of a few metres, while the inter-platoon spacing is assumed to be large so as to isolate the platoons from each other. From

the throughput point of view, the spacing should be as small as possible.

3) Join law: Under the join law, a leader or a free agent accelerates and merges with a platoon in front. It takes two platoons with an initial spacing and velocity mismatch to a final spacing equal to the safe following distance and zero velocity mismatches.

4) Split law: A following vehicle can use the split law to slow down and split from the original platoon to become a leading vehicle or a free agent. This control law takes a pair of vehicles from the intra-platoon spacing to the inter-platoon safe spacing. The design is similar to the join law.

5) Lane keeping law: The primary objective of this law is to maintain the lateral lane position of the vehicle. Several control laws have been proposed for lane keeping, such as the vision-based tracking controller [9].

6) Lane changing law: This control law guides the vehicle from one lane to the next. In the process, it is necessary to take into account the coupling between the longitudinal and lateral movements of the vehicle. This task seems to be the most demanding in terms of sensor requirements.

We summarise six operational modes, in real-time control, with each mode requiring two control laws, as listed in Table 1.

4 Statechart representation of hybrid controllers

In this section, the extended statechart modelling approach is applied to design the hybrid controller of the automated vehicles mentioned above.

4.1 Class diagram

The class diagram in Fig. 6 represents the static structure and object relations of the vehicle system. The *Vehicle* class has a composition relation (represented as a black diamond), with the *Vehicle Actuator* and *Vehicle Dynamics* classes, and an aggregation relation (represented as a white diamond), with the *Controller* and *Sensor* classes. The composition relation indicates that the composite is explicitly responsible for the creation and destruction of the contained objects. The aggregation relation is also an ownership relation but is somewhat weaker than the composition relation. The three attributes of the *Vehicle* class imply that an automated vehicle lies in either a leader, a follower, or a free agent mode. The *Controller* class consists of the *Coordination Controller* class with three manoeuvre operations and the *Regulation Controller* class with six feedback control operations and, thus, leads to a hybrid controller. Other relations in the diagram are associations, indicating loosely coupled classes that send messages to each other in order to collaborate.

Table 1: Operation modes and required control laws

Operational modes	Longitudinal laws	Lateral laws
Leader	Leader	Lane keeping
Follower	Follower	Lane keeping
Free agent	Leader	Lane keeping
Join	Join	Lane keeping
Split	Split	Lane keeping
Lane change	Leader	Lane changing

4.2 Statechart

The macroscopic statechart, shown in Fig. 7, represents the dynamic behaviour of the hybrid controller, and Fig. 8–11 show the hybrid system in more detail. After initialising the control system, the vehicle lies in one of either the leader, follower, or free agent modes and awaits link-layer commands. When the link layer sends commands to the coordination layer, the controller will analyse the platoon status and specify the necessary join, merge or lane change manoeuvre to perform (Fig. 8). Then, it generates the reference trajectories, including the desired jerk, acceleration, speed, spacing and yaw angle profiles, for the specified manoeuvre (Fig. 9). Before invoking the feedback control laws to perform a manoeuvre, the controller must make sure the requested action can be carried out safely. Thus, it performs certain safety checks, including checks of the vehicle status, given constraints and environment, as shown is Fig. 10. The status check contains general safety checks that will warn the driver if a malfunction occurs, such as a tire burst, engine breakdown or sensor failure. The constraint check deals with the limits of the engine

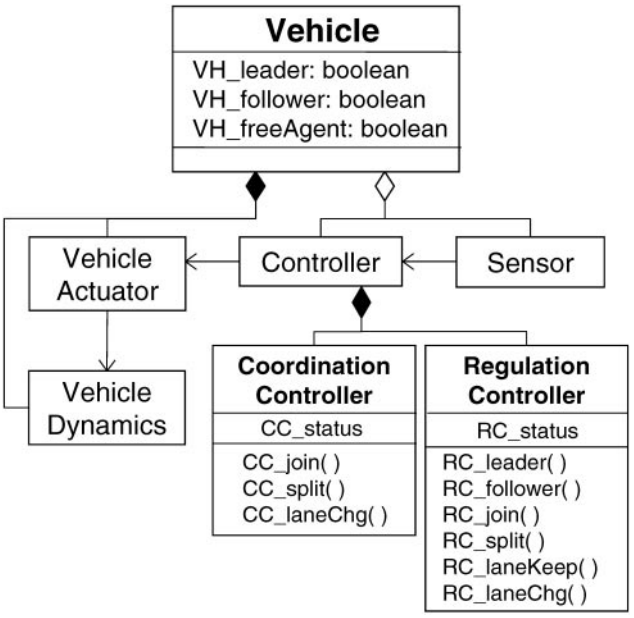

Fig. 6 *The class diagram of the automated vehicle system*

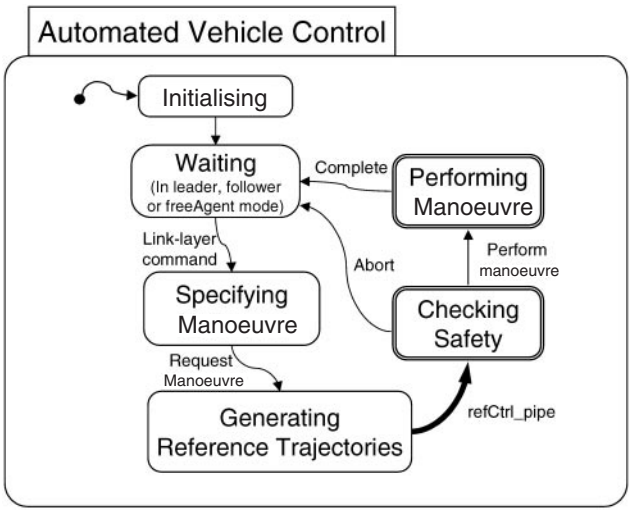

Fig. 7 *The macroscopic statechart of automated vehicle control*

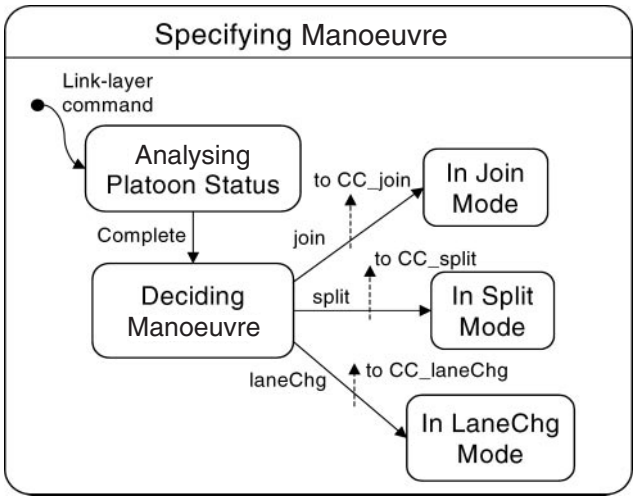

Fig. 8 *The statechart for specifying manoeuvre*

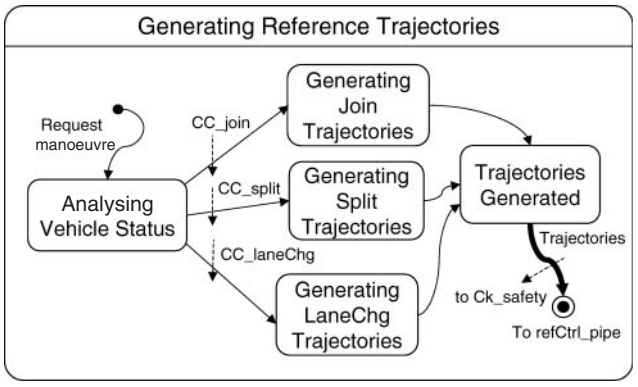

Fig. 9 *The statechart for generating reference trajectories*

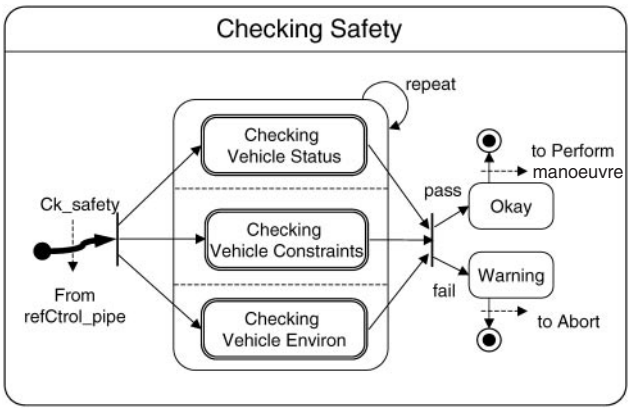

Fig. 10 *The statechart for checking safety*

capabilities and, if possible, the necessary jerk and acceleration bounds for passenger comfort. The environment check detects new vehicles in the vicinity to ensure collision-free operation while performing the manoeuvre. The checks are repeated whenever new sensor data come in. If any safety check fails, the controller will abort the manoeuvre. Otherwise, the controller will start to perform the requested manoeuvre. In *Performing Manoeuvre* state, shown is Fig. 11, the hybrid controller invokes the feedback control laws to perform the specified manoeuvre. The control laws use the reference trajectories and sensor signals to produce the throttle, brake and steering control inputs for the vehicle actuators. These control signals are directly applied to the actuators and, thus, manipulate

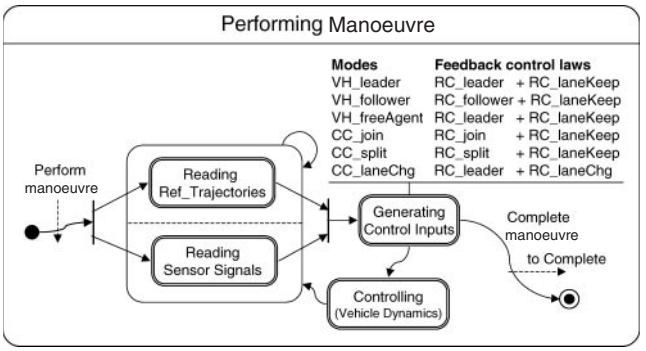

Fig. 11 *The statechart for performing manoeuvre*

the vehicle dynamics so as to track the desired trajectories for the specified manoeuvre. After completing the manoeuvre, the controller lies in the *Waiting* state to wait for the next link-layer command.

5 Verification issues of hybrid controllers

The aim of this paper is to present a statechart-based representation of the hybrid controllers for vehicle automation. Fig. 4 shows the hybrid controller comprising a discrete-event coordination controller and a discrete-time regulation controller. The verification method of the coordination layer could be found in [2], in which the logical correctness of the coordination controller was proved using a simple, finite state abstraction of the continuous dynamics. On the other hand, verification of the regulation controllers could be found in [3–9], in which mathematical transforms and root locus techniques were used to perform the analysis. However, it is possible for the regulation layer to produce certain undesirable behaviour, such as crashes, under severe disturbances. Thus, it is necessary to verify that the combined hybrid system does not result in vehicle crashes. To our best knowledge, at this time, no tools are available to analytically verify properties of such a complex hybrid system. In an AHS case especially, interaction between different vehicles, which are individually being controlled by the above mentioned hybrid control system, adds to the complexity of the verification. Owing to the lack of tools, simulation plays a very important role in the design of complex hybrid systems. There has been extensive work on the development of techniques for simulating general hierarchical hybrid systems [16]. A specific simulation package, SmartPath [20], is developed for simulation of automated vehicles in an AHS. Even though simulation cannot replace formal proof techniques, it can still provide valuable information about the system performance. In this paper, the present work is to model the hybrid controllers based on a statechart representation. Further verification and analysis of such complex hybrid systems would be investigated in the future.

6 Conclusion

This paper has presented a statechart-based approach towards the hybrid controller representation for automated vehicles in an AHS. For modelling hybrid systems, the statechart of the standard UML is extended to capture continuous time behaviour with a clear and natural representation. Although the dynamics of the overall hybrid systems are very complicated, our work illustrates a more general approach of the hybrid

controller representation for automated vehicles. Future work includes the study and comparison of the extended statechart from a theoretical point of view, the development of a powerful tool for automatic code generation and the fault tolerance ability with smart transition between the normal and automatic AHS driving. Also, analysis and verification of such a complicated hybrid system involving discrete event and continuous controllers is still an open question and could be further investigated.

7 Acknowledgements

This work was supported by the National Science Council, Taiwan, R.O.C., under grant NSC 92-2917-I-009-005 and in part by the Ministry of Economic Affairs under the Embedded System Software Laboratory in Domestic Communication and Optoelectronics Infrastructure Construction Project.

8 References

1 Varaiya, P.: 'Smart cars on smart roads: Problems of control', *IEEE Trans. Autom. Control*, 1993, **38**, (2), pp. 195–207
2 Hsu, A., Eskafi, F., Sachs, S., and Varaiya, P.: 'Protocol design for an automated highway system', *Discrete Event Dyn. Syst.: Theory Appl.*, 1993, **2**, (1), pp. 183–206
3 Huang, S., and Ren, W.: 'Longitudinal control with time delay in platooning', *IEE Proc. Control Theory Appl.*, 1998, **145**, (2), pp. 211–217
4 Stankovic, S.S., Stanojevic, M.J., and Siljak, D.D.: 'Decentralized overlapping control of a platoon of vehicles', *IEEE Trans. Control Syst. Technol.*, 2000, **8**, (5), pp. 816–832
5 Zhang, Y., Kosmatopoulos, E.B., Ioannou, P.A., and Chien, C.C.: 'Autonomous intelligent cruise control using front and back information for tight vehicle following maneuvers', *IEEE Trans. Veh. Technol.*, 1999, **48**, (1), pp. 319–328
6 Li, P., Alvarez, L., and Horowitz, R.: 'AHS safe control laws for platoon leaders', *IEEE Trans. Control Syst. Technol.*, 1997, **5**, (6), pp. 614–628
7 Shladover, S.E.: 'Longitudinal control of automotive vehicles in close-formation platoons', *ASME J. Dyn. Syst., Meas., Control*, 1991, **113**, pp. 231–241
8 Koumboulis, F.N., and Skarpetis, M.G.: 'Robust control of cars with front and rear wheel steering', *IEE Proc. Control Theory Appl.*, 2002, **149**, (5), pp. 394–404
9 Redmill, K.A.: 'A simple vision system for lane keeping'. Proc. IEEE Int. Conf. Intelligent Transportation Systems, Boston, MA, November 1997, pp. 212–217
10 Godbole, D.N., Lygeros, J., Singh, E., Deshpande, A., and Lindsey, A.E.: 'Communication protocols for a fault-tolerant automated highway system', *IEEE Trans. Control Syst. Technol.*, 2000, **8**, (5), pp. 787–800
11 Godbole, D.N., Eskafi, F., Singh, E., and Varaiya, P.: 'Design of entry and exit maneuvers for IVHS'. Proc. American Control Conference, Seattle, WA, July 1995, pp. 3576–3580
12 Cassandras, C.G., and Lafortune, S.: 'Introduction to discrete event systems' (Kluwer Academic Publishers, Boston, MA, 1999)
13 Charbonnier, F., Alla, H., and David, R.: 'The supervised control of discrete-event dynamic systems', *IEEE Trans. Control Syst. Technol.*, 1999, **7**, (2), pp. 175–187
14 Lee, J.S., and Hsu, P.L.: 'Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets', *IEEE Trans. Control Syst. Technol.*, 2004, **12**, (2), pp. 293–302
15 Lee, J.S., Zhou, M.C., and Hsu, P.L.: 'An application of Petri nets to supervisory control for human-computer interactive systems', *IEEE Trans. Ind. Electron.*, 2005, **52**, (5), pp. 1220–1226
16 Antsaklis, P., and Nerode, A.: 'Hybrid control systems: An introductory discussion to the special issue', *IEEE Trans. Autom. Control*, 1998, **43**, (4), pp. 457–460 (special issue on hybrid control systems)
17 Lygeros, J., and Godbole, D.N.: 'An interface between continuous and discrete event controllers for vehicle automation', *IEEE Trans. Veh. Technol.*, 1997, **46**, (1), pp. 229–241
18 Booch, G., Rumbaugh, J., and Jacobson, I.: 'The unified modelling language user guide' (Addison-Wesley, Reading, MA, 1999)
19 Harel, D.: 'Statecharts: A visual formalism for complex systems', *Sci. Comput. Program.*, 1987, **8**, pp. 231–274
20 Eskafi, F., Khorramabadi, D., and Varaiya, P.: 'SmartPath: An automated highway system simulator'. Tech. Report, Institute of Transportation Studies, University of California, Berkeley, 1994