Contents

CHAPTER 10

Stop and Go Adaptive Cruise Control: A Case Study of Automotive Cyber-Physical Systems

Sasan Vakili

McMaster Centre for Software Certification, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Neeraj Kumar Singh

INPT-ENSEEIHT/IRIT, University of Toulouse, Toulouse, France

Mark Lawford

McMaster Centre for Software Certification, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Alan Wassyng

McMaster Centre for Software Certification, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Ben Breimer

McMaster Centre for Software Certification, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

CONTENTS

Automotive Cyber Physical Systems (Auto-CPS) are safety critical systems in which failure can lead to grave consequences, such as financial loss, severe injuries, and even loss of life. Auto-CPS involve tightly coupled interaction of computational units and physical systems that involves interplay between embedded systems, control theory, real-time systems and software engineering. Increases in software related recalls of vehicles are driving demand for methods to cost effectively produce safe, secure, and reliable cars with more than 100 electronic control units (ECUs) communicating over vehicle networks. Since formal methods have played a significant role for developing a safe and dependable systems, the role of formal methods in developing Auto-CPS should be taken into account. In this work we will demonstrate the benefits and limitations of formal methods for Auto-CPS by considering a relatively new driver assistance feature.

Stop and Go Adaptive Cruise Control (ACC+) is an extension of Adaptive Cruise Control (ACC) that assists drivers by regulating the speed of the driver's vehicle relative to the vehicle it is following for speeds from 0 to a drive set maximum cruising speed. In this paper, we present the formal verification of a practical, robust ACC+ design using differential dynamic logic (dL) to formally state environmental assumptions and prove safety goals, including collision freedom. The verification is done in two stages. First, we identify the invariant required to ensure the safe operation of the system and we formally verify that the invariant preserves the safety property of any system with similar dynamics. This procedure provides a high level abstraction of a class of safe solutions for ACC+ system designs. Second, we show that our ACC+ system design is a safety preserving refinement of the abstract model. The safety of the closed loop ACC+ system is proven by verifying bounds on the system variables using the KeYmaera verification tool for hybrid systems. The work provides a method that could be used to verify more complicated ACC+

controller designs that are optimized for fuel economy, passenger comfort, etc. We use our formal specification to create a Matlab/Simulink implementaion to validate the performance of the proposed designs and compare our design with other formally verified ACC designs from the literature.

10.1 INTRODUCTION

A promising and challenging application area for formal methods in the automotive domain is specifying and designing software for embedded systems since these types of systems growing exponentially in complexity [4]. Software has become a decisive factor in the automotive industry. Increasing demands for high quality, safety requirements, and the shortcomings of the informal techniques applied in traditional development have motivated the examination of semi-formal or formal methods to facilitate a higher degree of automation and tool support for verification and validation activities to insure safety.

Every year, car crashes result in the loss of thousands of lives, and permanent disabilities, resulting in annual costs of billions of dollars only in United States [31]. Although the majority of these car crashes are due to human error, failure in hardware or software components can lead to accidents and unduly risk human life [19]. While hardware failures are typically some kind of *random failure* that is caused by different wear effects such as corrosion, thermal stressing, *etc*, software failures are *systematic failures* that may be introduced by human error during the system development, and these failures always appear in the same circumstances, until the error is removed. However, it is difficult to predict the occurrence of systematic software related failures or detect all of them by classical means such as testing and inspections [18].

Formal verification is an effective approach to help ensure the safety and reliability of complex hybrid systems that can provide an additional level of confidence. This work contributes to the formal verification of automotive hybrid systems by using *dynamic logic* ($\mathrm{d}\mathcal{L}$) to analyze the correctness of a high level ACC^+ design. The verification method used assures safety of the system is robust with respect to variations in the plant models. Formal development and verification of hybrid systems using dynamic logic provides satisfaction of safety and performance requirements if the models used correctly represent the system. Using parametric constraints, we find a region of safe operation for a continuous controller when we have an upper bound limit on response time in the presence of disturbances and uncertainties. In addition to the advantages of formal verification given above, making system descriptions more precise can serve to expose problematic parts in the requirements. Based on the formal model of the system, we want the analysis techniques to establish the system correctness to be consistent with requirements. The main contributions of the paper are:

1. a new high-level design for ACC^+.
2. formalization of the ACC^+ requirements using $\mathrm{d}\mathcal{L}$.

3. formal verification of the new design's safety properties using the KeYmaera tool [24].

The outline of the remainder of the paper is as follows. Section 2 presents related work. A brief outline of ACC$^+$ and the tools and techniques used in the verification of the system are described in Section 3. Section 4 presents a high level design of an ACC$^+$ controller, and Section 5 describes the formal verification of the ACC$^+$ design. Section 6 concludes the paper and discusses future work.

10.2 RELATED WORKS

Automotive control is a wide and interesting area that has been studied by academic and industrial researchers in an effort to minimize the risk, and to improve the safety of driving. Since these kinds of systems deal with safety of humans, even a small error or mistake in the design of these systems can lead to irreparable harm. Therefore, sufficient assurance is necessary before deployment of any system.

Guven et al. [9] presented a low-cost, real-time driver-in-the-loop vehicle simulator that was used to analyze ACC behaviour for highway traffic. Several papers [29, 8, 3, 17] reported work on the simulation of ACC. However, these simulations are not enough to guarantee that the tested system is safe and collision free in all traffic conditions.

Our ACC$^+$ design is a hybrid system. Hybrid systems integrate both continuous and discrete dynamics, bringing together several research fields to address challenging problems in order to demonstrate safe operations. Logic plays a significant role in formal verification of hybrid systems from reachability analysis to undecidability in theory and practice.

Several approaches have been proposed in literatures to verify safety properties of hybrid systems. An inductive method proposed in [1] based on the PVS theorem prover to verify the required safety properties of the parallel hybrid systems. SMT solvers have been used to verify safety properties and time-bounded reachability of a class of "reasonable" parametric linear hybrid automata [6]. Stursberg *et al.* [27] presented a counterexample-guided verification approach based-on a model checker to identify the unwanted behaviours in the verification of a cruise control system, where a sequence of abstractions was used to reduce the computational cost. Jairam et al. [10] present verification of a MEMS based ACC system using simulation and semi-formal approaches, where they have used Matlab Simulink to develop the case study and validate their system using a transformation based approach. An interesting pieces of work is presented in [5], where a theoretically ACC system is described by process algebra (*timed distributed pi-calculus*) to analyse the informal requirements of the system and some properties such as deadlocks are then verified using the Mobility Workbench model checker.

Platzer *et al.* proposed dynamic logic and proof calculus for verifying hybrid systems [21, 23]. Dynamic logic considers continuous evolutions between discrete behaviour via transitions between the states. In the past few years Loos *et al.* have published several papers applying KeYmaera to ACC [12, 13]. All these papers address fundamental principles of ACC, including important safety properties in various scenarios. However, none of them provide a feasible solution for implementation purposes. For instance, [12] discussed formal verification of ACC considering the required following distance for avoiding collision when there are arbitrarily many cars driving on a highway, including the case when another car enters the lane. Another paper, [13], proposed ACC modelling based on different acceleration choices for different modes of operation using various conditions, however, it has overlap in the modes that causes thrashing due to improper guard conditions defined in the paper. The mode thrashing has the potential to result in changes of acceleration that would be unacceptable in terms of driver comfort and fuel efficiency. The second and fourth controller mode in [13] is unreachable regardless of the plant model. Moreover, [13] proposed an acceleration formula for the third controller mode using a square root term involving variables such as communication time (τ). However, the optimal τ assumed as the maximum communication time (3.2 seconds) is unrealistic for a real time application. These two proposed solutions for ACC have not taken any desired set point velocity or distance into account, which is also required to consider during formalizing a complete system that could serve as a potential ACC system. In another paper Aréchiga *et al.* [2] proposed a PID controller based on the previous paper [12] to maintain a desired distance between two successive vehicles when the host vehicle follows a lead vehicle, describing acceleration in terms of position and velocity of vehicles. However, a large desired following distance is attained in their controller design which makes the system unrealistic. The system gets into an unsafe region if a small desired following distance is considered. The problem with this system is that the specified operating regime for the controller is in the unsafe boundary for small set point values and the controller will not take any action in some unsafe scenarios. Therefore, only unrealistically large following distance set point values can be used in order to satisfy the safety condition.

Our motivation is to provide a complete solution for a practical, generic ACC$^+$ system design that guarantees the safety properties outlined in [12, 2, 13] while incorporating practical ACC$^+$ design requirements such as headway reference tracking and respecting the user specified maximum velocity constraint, both of which are missing from the works of Loos *et al.* Our proposed solution allows the host vehicle to maintain a desired velocity when there is no slower lead vehicle or obstacle, and to safely approach a slower lead vehicle to a desired safe following distance. Moreover, we investigate the system's safety critical behavior as a separate mode of operation used to guarantee safety and collision freedom properties.

10.3 PRELIMINARIES

10.3.1 ACC$^+$

In order to understand an ACC$^+$ system, one should first understand prior iterations such as cruise control (CC) and adaptive cruise control (ACC) systems. The main function of ACC is to maintain a desired speed that is set by the driver unless a slower "lead vehicle" is encountered ahead of the driver's vehicle - the "host vehicle" for the ACC system. For example, if the current speed of a lead vehicle is slower than the speed of the host vehicle, then the ACC system starts to control the current speed of the host vehicle to maintain a desired safe distance between lead vehicle and the host vehicle. Automatic adjustment of the host vehicle's acceleration allows the host vehicle to adjust its speed according to the traffic conditions without driver intervention. Any required braking action carried out by an ACC system will typically not exceed 30% of the host vehicle's maximum deceleration. When stronger deceleration is needed, driver is warned by an auditory signal and a warning message is displayed on a driver information screen. The driver can override the ACC system at any time to take back control of the vehicle. The ACC system must guarantee that it will always behave correctly and safely while respecting rules regarding passenger comfort (i.e. trying to avoid excessive changes in acceleration). Consideration of traffic conditions can be included in an ACC system design to help decrease traffic congestion by trying to provide a smooth flow of traffic. An ACC system has a minimum threshold speed (e.g. 30 km/h) below which it stops operating, hence an ACC system does not deal with stop and go traffic [26, 16].

Stop and Go Adaptive Cruise Control, also known as Adaptive Cruise Control Plus (ACC$^+$), is a system that operates at all velocities greater than or equal to 0 km/h. It is an extension of the ACC system that is basically a superset of the features found in ACC. In particular ACC$^+$ is designed to provide controllability in very low speed driving scenarios. In [30] the development of an ACC$^+$ system is discussed, including some of the challenges that arise from low speed driving such as smaller inter-vehicular spacing and more frequent changes in velocity. An ACC$^+$ system must obtain information about its environment such as the speed of the lead vehicle, the user's requested speed, what constitues a safe distance between the host vehicle and the lead vehicle, *etc.*, in order to meet its requirements. This can be achieved by using a set of sensors that monitor the environment at sufficiently high sampling rates to capture the continuous behaviour of hybrid system in sufficient detail. For example, may ACC$^+$ system use a frequency modulated continuous wave (FMCW) doppler rafar sensor mounted on the front of the vehicle to measure the distance to the lead vehicle and its relative velocity [25]. Since the system must both accelerate and de-accelerate the host vehicle based on the information obtained from the environment, an ACC$^+$ system must be able to adjust the throttle and/or brake using appropriate control signals.

Uncertainty is another fact that cannot be ignored in the design of this system. No mathematical model can represent the exact physical system [7]. Therefore, different types of uncertainty should be taken into account during the process of controller design. Considering all of these aspects makes the system complex and difficult to assure safety and correctness. The goal of this work is to design a robust ACC$^+$ system that provides sufficient assurances of safety for typical operating conditions.

10.3.2 Differential Dynamic Logic (d$\mathcal{L}$)

Differential dynamic logic (d$\mathcal{L}$) is a first-order dynamic logic for specification and verification of hybrid systems. A program notation is used to describe hybrid systems as hybrid programs, with symbolic parameters being used during the verification process in dynamic logic. A free variable sequential composition proof calculus with real arithmetic and quantifier elimination allows deductive verification of hybrid programs [21, 23, 22].

Symbolic parameters are represented by a set of logical variables in the first-order logic, while dynamic logic describes the continuous behaviour of a system. This logic can be used to verify the operation of a system with discrete and continuous state transitions by introducing hybrid programs with discrete assignments and differential actions and then applying a deductive method rather than using abstractions and exhaustive state space exploration as is typically done in model checking approaches [20]. The limited knowledge of d$\mathcal{L}$ needed to understand this paper is summarized below. More detail descriptions are available in [21, 22].

Dynamic logic (d$\mathcal{L}$) consists of nonlinear real arithmetic, real valued quantifiers, and modal operators, such as $\langle\alpha\rangle$ or $[\alpha]$ for expressing reachable state conditions during system execution, where α presents the continuous evolution of a system. A set of logical variables V, a signature Σ, a set of real valued function and predicate symbols are used to define the well-formed terms and formulas that are given as follows:

$$\theta ::= x \mid f(\theta_1, ..., \theta_n)$$

where $\theta_1, ..., \theta_n$ are terms, f is a function symbol of arity n, and x is a real-valued constant symbol.

$$\phi, \psi ::= p(\theta_1, ..., \theta_n) \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \forall x \phi \mid \exists x \phi$$

where ϕ and ψ are first-order formulas, θ_i are terms, p is a predicate symbol of arity n, and $x \in V$ is a logical variable.

Hybrid programs consist of discrete jump sets, systems of differential equations and a control structure. The discrete transitions assign values to the state variables, and the differential equations are used to express a continuous dynamic evolution of the system, which may change from one discrete state to another. The control structure plays an important role for combining the discrete and continuous transitions using regular expression operators, such as ($\cup$, *, ;). The grammar for designing the hybrid programs is given as follows:

$$\alpha, \beta ::= \ x_1 := \theta_1, ..., x_n := \theta_n \mid x_1' = \theta_1, ..., x_n' = \theta_n \& \chi \mid ?\chi \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

where α and β are hybrid programs, θ_i are terms, $x_i \in \Sigma$ are state variables, and χ is a formula of first-order logic. $x_1 := \theta_1, ..., x_n := \theta_n$ shows a discrete jump, in which θ_i assigns to state variables x_i. $x_1' = \theta_1, ..., x_n' = \theta_n \& \chi$ presents a list of differential equations for describing dynamic behaviour with additional first-order constraints χ. $?\chi$ and $\alpha \cup \beta$ are used to test the state variables and represent nondeterministic choice, respectively. $\alpha; \beta$ and α^* present sequential composition and nondeterministic repetition, respectively. Dynamic logic (d$\mathcal{L}$) can be used to design other structures by combining the control structure operators ($\cup$, *, ;) with $?\chi$, such as in conditional statements like **if** χ **then** α **else** β, **while** χ **do** α. Formulas of dynamic logic (d$\mathcal{L}$) based-on first-order logic together with some modal operators ($<\alpha>$ or $[\alpha]$) are defined as follows:

$$\phi, \psi ::= p(\theta_1, ..., \theta_n) \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \forall x \phi \mid \exists x \phi \mid [\alpha]\phi \mid <\alpha> \phi$$

where ϕ, ψ are dynamic logic (d$\mathcal{L}$) formulas, θ_i are terms, p is a predicate symbol of arity n, $x \in V$ is a logical variable, and α is a hybrid program. The syntax of dynamic logic (d$\mathcal{L}$) allows real arithmetic predicate expressions, negation, conjunction, disjunction, implication, universal and existential quantification, and modalities to express the validity of formula ϕ for any terminating execution of hybrid program α ($[\alpha]\phi$) or at least one terminating execution of hybrid program α ($<\alpha> \phi$).

10.3.3 Verification Tool: KeYmaera

KeYmaera [24] is a hybrid verification tool integrated with an automated and an interactive theorem prover to formalize and verify hybrid systems. It supports dynamic logic (d$\mathcal{L}$), and combines different methods, such as deductive logic, real algebraic, and computer algebraic rules. Moreover, KeYmaera also supports nonlinear discrete jumps, nonlinear differential equations, differential-algebraic equations, differential inequalities, and nondeterministic discrete or continuous input for hybrid systems to express the functional behaviours. KeYmaera allows decomposition of hybrid system specifications into symbolic form and into subsystems to simplify the proof strategy. However, a bottom-up approach employing compositional verification allows KeYmaera to verify large, complex systems by proving the required properties of the sub-systems and then the main system.

10.4 A HIGH LEVEL SAFETY CONCEPT ABSTRACTION FOR ACC$^+$ SYSTEMS

ACC$^+$ systems can be formalized using *differential dynamic logic* (d$\mathcal{L}$) to state and prove safety properties and performance requirements by capturing the system constraints together with the desired behaviours and controller designs. A high level conceptual safety design of ACC$^+$ is proposed in this section, where we consider that the host vehicle is equipped with ACC$^+$, and the host vehicle follows a lead vehicle in the same lane. We use a high level conceptual model of the ACC$^+$ system to formalize an abstraction of the system requirements to satisfy the desired safety properties. According to [12, 2], collision-freedom for these kinds of systems can be achieved if and only if there is always a safe distance between two successive vehicle. This distance, which we will denote by sc_{gap}, can be derived from Newton's formula of motion as in Eq. 10.1, where B is the absolute value of maximum deceleration achieved by maximum brake force, and v_l and v_h are lead and host vehicles' velocities respectively.

$$sc_{gap}(v_l, v_h) = \frac{v_h^2 - v_l^2}{2 \times B} \tag{10.1}$$

The length of $sc_{gap}(v_l, v_h)$ should be such that the host vehicle can fully stop at the rear end of the lead vehicle or end of $sc_{gap}(v_l, v_h)$ in the worst case scenario when the lead vehicle may itself be suddenly using the same maximum brake force to come to a full stop. In the case when the relative distance between the two vehicles is less than or equal to this safe distance, the host vehicle has no choice but to use its maximum braking power to exit the critical zone in order to make the system collision free. This fact is critical to the safe, collision-free operation of any ACC or ACC$^+$ design.

In addition, the system uses sensors to provide required values for the control system; however, there is some lag associated with acquiring sensor reading, the controller needs some time to react to any new sensor values, and the actuators take some time to react. Therefore, a safety margin should be taken into account related to the maximum delays in the system. This extra padding distance can be determined by the following formula (Eq. 10.2) according to [12].

$$margin_{sc_{gap}}(v_h) = \left(\ \tfrac{A_{max}}{B} + 1\ \right)\left(\ \tfrac{A_{max}}{2} \times \epsilon^2 + \epsilon \times v_h\ \right) \tag{10.2}$$

Here A_{max} is the maximum acceleration of the host vehicle and ϵ is the worst case delay time, which is close to zero. Eq. 10.2 is considered as the worst case scenario where the host vehicle is traveling with max acceleration (A_{max}) when the ACC$^+$ system requests the maximum negative acceleration B. The host vehicle will continue to accelerate at A_{max}, increasing its velocity v_h for ϵ seconds before it starts to decelerate at $-B$. Therefore, the extra distance given in Eq. 10.2 is required for acceleration $-B$ to return the host vehicle to what was its initial velocity, v_h, when the negative deceleration was first requested. Consequently, $margin_{sc_{gap}}(v_h)$ is the total of these two distances that the host vehicle travels during the ϵ delay. Thus the ACC$^+$ system can react safely if the relative distance between host and lead vehicle ($d_{gap} = x_l - x_h$) is always greater than the sum of $sc_{gap}(v_l, v_h)$ and $margin_{sc_{gap}}(v_h)$ as in Eq. 10.3.

$$d_{gap} > sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h) \tag{10.3}$$

Therefore, the High Level Safety Concept Abstraction for ACC$^+$ can be specified as shown in Fig. 10.1. This figure depicts an independent safety system that intervenes only when necessary. This system monitors the relative distance to the lead vehicle (d_{gap}) and sets the host vehicle acceleration a_h to $-B$ whenever the relative distance is less than or equal to safety distance ($d_{gap} \leq sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$). Effectively it activates a *Safety_ Critical* mode that applied the maximum brake force. Four components have been considered for this purpose: a *guard condition* block, a switching block, *safety_ Critical*, and *Other*. The *guard condition* block checks the validity of the safety condition and the switching block changes the active mode from *Other* (normal ACC$^+$) functionalities to *Safety_ Critical* in critical cases when $d_{gap} \leq sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$ becomes valid. A tabular representation of this decision making structure is shown in Table 10.1, where *Other* is used to consider all other behaviour ACC$^+$ could have in different scenarios and *Safety_ Critical* is used to apply maximum brake.

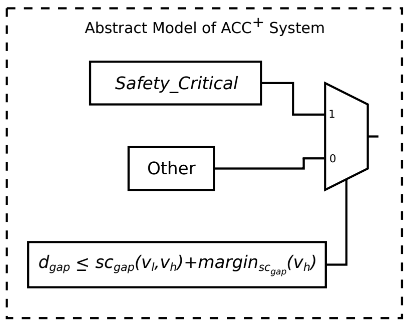

Abstract Model of ACC$^+$ System
Safety_Critical
1
Other
0
$d_{gap} \leq sc_{gap}(v_l,v_h)+margin_{sc_{gap}}(v_h)$

FIGURE 10.1: High level abstract conceptual design block diagram

	a_h	*Mode*
$d_{gap} \leq sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$	$-B$	*Safety_ Critical*
$d_{gap} > sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$	$[-B, A_{max}]$	*Other*

TABLE 10.1: Decision making structure of abstract ACC$^+$

10.4.1 Verification

Since some of the system parameters come from the environment or are yet to be determined by a more detailed design, we need to define symbolic constraints for some parameters like acceleration of the vehicles. Before doing this we first define the state variables of the host and lead vehicles that will be used to model their continuous behaviour:

$$host = (x_h, v_h, a_h) \tag{10.4}$$

$$leader = (x_l, v_l, a_l) \tag{10.5}$$

where x_h is position, v_h is velocity, and a_h is acceleration of the host vehicle, and x_l is position, v_l is velocity, and a_l is acceleration of the lead vehicle. These variables from the tuples can then be used to specify the dynamics of the real-time system, where the relationships between position, velocity and acceleration are $x'_h = v_h$ and $v'_h = a_h$ for the host vehicle, and $x'_l = v_l$ and $v'_l = a_l$ for the lead vehicle.

The velocity of the host (lead) vehicle changes continuously according to the current acceleration of the host (lead) vehicle. We assume the maximum acceleration for both the host and lead vehicles is $A_{max} > 0$, and similarly the maximum deceleration due to braking with the maximum braking force is $-B$ where $B > 0$. Therefore,

$$-B \leq a_h \leq A_{max} \ \ \& \ \ -B \leq a_l \leq A_{max} \tag{10.6}$$

The complete formalization of our abstract ACC$^+$ is presented in **Model 1**. The model contains both discrete and continuous dynamic behaviours. **Model 1** can be derived in a similar fashion to [12], where Loos *et al.* also define an abstract model for an autonomous vehicle. However, **Model 1** presented here is simpler and more abstract than the model in [12]. The *Local Lane Control* of the ACC system in [12] always sets the acceleration of the host vehicle to zero in the case that its velocity is zero. Thus once stopped, the ACC system in [12] remains stopped regardless of the behaviour of the lead vehicle. Also, *Local Lane Control* in [12] chooses a nondeterministic brake value within a particular range for the safety critical situation, which makes the system more complicated than a safety concept abstraction. Despite its complexity, *Local Lane Control* [12] is more realistic than a basic safety concept abstraction since it might not always be possible to achieve $-B$, for example when the road is wet.

The host and lead vehicles can repeatedly choose an acceleration from the range $[-B, A_{max}]$ in **Model 1**. This behaviour is specified by the nondeterministic repetition $*$ in line (1). The host and lead vehicles operate in parallel as defined in (2). The lead vehicle is free to use brake or acceleration at any time so a_l is assigned non-deterministically in (3), and the model continues if a_l is within its accepted range $[-B, A_{max}]$.

The host vehicle's movement depends on the distance between the host vehicle and the lead vehicle. The most crucial functionality of ACC$^+$ is formalized as successive actions to capture the decision on entering the safety critical mode as the last action in (4) before the system's continuous state is updated. The *safety critical following distance* ($sc_{gap}(v_l, v_h)$) and the *extra safety margin for delays* ($margin_{sc_{gap}}(v_h)$) are calculated in (5). The last line in (5) assigns the relative distance to d_{gap} . The host vehicle can choose any arbitrary acceleration value in the valid range $-B$ to A_{max} for the *Other* mode in (6) to capture all dynamic behaviours of possible ACC$^+$ system designs. The safety requirement that the system applies maximum brake force when the host vehicle is within the safe following distance is formalized as the overriding action of the *Safety_ Critical* mode in line (7). The continuous state of the system then evolves over time that is measured by a clock variable t. The sampling time of the system has been considered as the delay of the system $t \leq \epsilon$ where slope is considered as $t' = 1$. Therefore, the system is piecewise continuous and the physical laws for movement as formalized by simplified versions of Newton's formula, are contained in line (8).

Model 1: Formalization of abstract model for ACC$^+$ systems

$$ACC^+ \equiv (Vehicle; Drive)^* \quad (1)$$

$$Vehicle \equiv host \,||\, leader; \quad (2)$$

$$leader \equiv a_l = *; ?(-B \leq a_l \leq A_{max}) \quad (3)$$

$$host \equiv Calc\_sc_{gap};\ Other;\ Safety\_\ Critical; \quad (4)$$

$$\begin{aligned} Calc\_sc_{gap} \equiv\ & sc_{gap}(v_l, v_h) := \tfrac{v_h^2 - v_l^2}{2 \times B}; \\ & margin_{sc_{gap}}(v_h) := (\tfrac{A_{max}}{B} + 1)(\tfrac{A_{max}}{2} \times \epsilon^2 + \epsilon \times v_h); \\ & d_{gap} := x_l - x_h; \end{aligned} \quad (5)$$

$$Other \equiv a_h := *; ?(-B \leq a_h \leq A_{max}); \quad (6)$$

$$\begin{aligned} Safety\_\ Critical \equiv\ & \text{if } \big(d_{gap} \leq sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)\big) \text{ then} \\ & a_h := -B \\ & \text{fi}; \end{aligned} \quad (7)$$

$$\begin{aligned} Drive \equiv\ & t := 0; (x_h' = v_h \wedge v_h' = a_h \wedge x_l' = v_l \wedge \\ & v_l' = a_l \wedge t' = 1 \wedge v_h \geq 0 \wedge v_l \geq 0 \wedge t \leq \epsilon) \end{aligned} \quad (8)$$

With the system dynamics specified, we can now use the KeYmaera [24] tool to verify the required collision-freedom safety property.

Property 1: *If the host vehicle is following at a safe distance behind the lead vehicle, then the vehicles will never collide in any operation when the host vehicle controllers follow the defined dynamics given by the safety constraints.*

In KeYmaera this property will take the form:

$$\text{Controllability Condition} \rightarrow [\text{Abstract ACC}^{+}] \;\; x_h < x_l \tag{10.7}$$

The controllability condition will be given below in equation (10.8). We now explain how we will arrive at the appropriate precondition for the safety property. To complete (10.7), we must establish a precondition that says that the host vehicle is behind the lead vehicle and both vehicles are moving in a forward direction. The relation (10.7) indicates that for all iterations of the hybrid program in **Model 1** the position of the host vehicle is always less than lead's vehicle's position ($x_h < x_l$) if the given controllability condition is satisfied. In other words, the relative distance between the vehicles is always greater than zero ($d_{gap} > 0$) if the precondition holds. One of the most important condition is the safe distance formula, which is an invariant during the proof of this hybrid program. This condition can be considered as a controllability property and must always be satisfied by every operation of the ACC^{+} system.

10.4.2 Controllability

The controllability formula states that for every possible evolution of the ACC^{+} system, it can satisfy the safety property by applying maximum brake before it has passed the *Safety_ Critical* distance. The vehicle is controllable if there is enough distance in order to fully stop the car by the rear end of lead vehicle or it exits the critical zone. The assumption is that both vehicles only move forward (i.e. their velocity is greater than or equal to zero). Therefore, the ACC^{+} system will be safe if it can satisfy condition (10.8), which is an invariant for the defined systems dynamics of **Model 1**. This controllability property in condition (10.8) is a safety concept invariant not only for ACC^{+} systems, but also for any kind of system with similar continuous motion dynamics.

$$x_l > x_h \wedge v_h^2 - v_l^2 < 2 \times B \times d_{gap} \wedge v_l \geq 0 \wedge v_h \geq 0 \tag{10.8}$$

An important fact in this verification is that there must be required distance to be physically possible to stop the host vehicle by the rear end of any obstacle appearing in front of the host vehicle. This fact has been mathematically indicated in (10.8) as $v_h^2 - v_l^2 < 2 \times B \times d_{gap}$. The system checks whether it can satisfy the safety property in case of detecting any obstacle and once it enters the critical zone, it uses maximum brake until the safety property holds again.

This model has been written in the KeYmaera theorem prover [24] and the required safety property (10.7) has been successfully proven. In this abstract model of an ACC^{+} system, we considered a viable range of accelerations for the host vehicle that admits a variety of desired behaviour for a concrete ACC^{+} system in different scenarios. The focus here is the desired behaviour of any of these concrete models in the safety critical case required to guarantee the safety requirement of collision freedom. In the next section, we will refine this system with respect to other requirements to create a more realistic concrete ACC$^+$ design, the safety of which has already been proven if we can show the new design refines this Abstract ACC$^+$ safety concept.

10.5 REFINEMENT OF THE SAFETY CONCEPT INTO A PRACTICAL ACC$^+$ MODEL

An ACC$^+$ design requires information about the host vehicle's continuous state (velocity, acceleration, etc.), as well as information about the presence and behaviour of the lead vehicle. While the most important requirement for ACC$^+$ systems is to safely adjust the host vehicle's speed in the presence of a lead vehicle, some additional functional requirements and assumptions have to be considered in the design of ACC$^+$ systems. Assumptions can help to make the design more reliable and practical. Also, understanding additional functional requirements can allow us to scope the design and verification effort.

Assumptions:

1. The ACC$^+$ system will never be operating when the vehicle is moving backwards ($velocity < 0$).
2. The driver is responsible for steering the host vehicle in a safe manner.
3. It is assumed that the maximum range of the sensors for detecting objects in front of the host vehicle (d_{range}) is always greater than the safety gap obtained in the previous section (Section 10.4):

$$d_{range} > sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h).$$

4. Errors will be detected by a separate subsystem, a Fault Detection System, that will alert the driver to intervene in the case of a fault.

Given Requirements:

1. The user has the ability to override the ACC$^+$ system settings such as desired velocity v_{set} and desired headway h_{set}, at any point in the system's operation except in safety critical cases.
2. The accessible parameters of the ACC$^+$ system, such as desired velocity v_{set} and desired headway h_{set}, should be restricted to an acceptable range in order to meet the assumptions and limitations of the design.
3. The ACC$^+$ system must regulate the velocity of the host vehicle to maintain the user's expected velocity in the absence of a slower lead vehicle.

4. The ACC$^+$ system must slow down the host vehicle's velocity and maintain the desired headway when approaching a slower lead vehicle.
5. The acceleration of the system must be restricted to a comfortable range. Therefore, rapid de-acceleration should not be applied during the normal operation of the ACC$^+$ system.
6. The ACC$^+$ system should return the operation of the vehicle to the user in the presence of any failure in the system or when the throttle/brake is touched by the user.

Among all the requirements, we consider the implementation of the first to fifth one in our design. The third and fourth requirements, which are not typically discussed in other related works such as [12, 13], play a major role in our ACC$^+$ design. The restriction on v_{set}, as described by the second requirement, is derived in sections 10.5.1 and 10.5.2. The required restriction on h_{set} can be derived in a similar fashion to v_{set}. The sixth requirement is not directly addressed in our work. It can be designed in a separate block by using fault diagnosis techniques as in [15]. The ACC$^+$ system controls the speed of the host vehicle according to the different scenarios that are considered during the high level design. Fig. 10.2 depicts a high level design of the ACC$^+$ that contains four components: a *low-level (continuous) controller*, an extended *finite state machine* (FSM), a *sensor*, and the *host vehicle*. We consider the fifth component, the *lead vehicle*, as being external to the ACC$^+$ system. All the components of the ACC$^+$ system are connected by arrows that represent the system data flow. Thus this block diagram shows the flow of information that is required to design the ACC$^+$ system, providing the relationship between the ACC$^+$ subsystems and the lead vehicle. *Mode* is the value of the current state of the FSM that is used by the low-level controller to select a particular continuous controller. The value of *Mode* belongs to the set {*Cruise*, *Follow*, *Safety_ Critical*}. Signal v_{ref} is a reference signal for the target velocity for the continuous controller selected inside the low-level controller. A list of the other symbols for describing vehicle behaviour is given in Table 10.2.

10.5.1 Controller Modes

There are three main operational modes of our ACC$^+$ design (see Fig.10.4). These modes are:

Cruise which implements standard cruise control system (CC) when no lead vehicle is detected or the lead vehicle exceeds the desired maximum velocity of the host vehicle (v_{set}) and is outside of the safety critical region.

Follow which tries to match the lead vehicle's velocity at distance $h_{set} \times v_l$, and

Term	**Definition**
Specification Terms	
v_h	velocity of the host vehicle
v_l	velocity of the lead vehicle
a_h	acceleration of host vehicle
a_l	acceleration of lead vehicle
d_{gap}	relative distance between the host vehicle and lead vehicle
Controller Terms	
v_{set}	desired velocity of the host vehicle
B	absolute value of deceleration achieved by maximum brake force, which depends upon the current vehicle weight and road conditions
h_{set}	desired following time gap between two successive vehicle (headway)
ϵ	maximum response delay from any actuators (ie. engine, brake etc.)
$f_{gap}(v_l, v_h, a_h)$	is the distance it takes for the host vehicle to match the lead vehicle's velocity and be following at the desired headway h_{set} using acceleration a_h
$sc_{gap}(v_l, v_h)$	is the distance at which ACC$^+$ system switches into safety critical mode

TABLE 10.2: Terms used in ACC$^+$ Specification and Controller Design

Safety_Critical where the vehicle has to apply maximum braking force to avoid a collision as discussed in the previous section (Section 10.4).

Fig. 10.3(a) and Fig. 10.3(b) show the headway diagrams describing the possible scenarios. The first mode is similar to a conventional cruise control system (CC) that regulates the speed of the host vehicle to the desired set point (v_{set}) within acceleration limits based on the requirements such as comfort and fuel efficiency. If the host vehicle detects a leader or other object, the system determines whether or not the sensed object is going faster than v_{set}. If the lead vehicle is travelling faster than v_{set} and is outside of the safety critical zone ($d_{gap} > sc_{gap}(v_l, v_h)$) then the ACC$^+$ system will not change its operating mode.

The second mode, *Follow*, becomes active when the host vehicle follows a slower lead vehicle outside the safety critical zone. In this situation, the objective is to maintain the desired headway gap $h_{set} \times v_l$ while diverse aspects such as driver comfort, fuel economy, *etc.*, are considered. When a slower lead vehicle is present, the goal is to reduce the host vehicle's velocity as it approaches the lead vehicle, matching the lead vehicle's velocity when the gap closes to the desired headway $h_{set} \times v_l$. To achieve this behaviour the system picks a negative acceleration for the host vehicle ($a_h < 0$). An additional

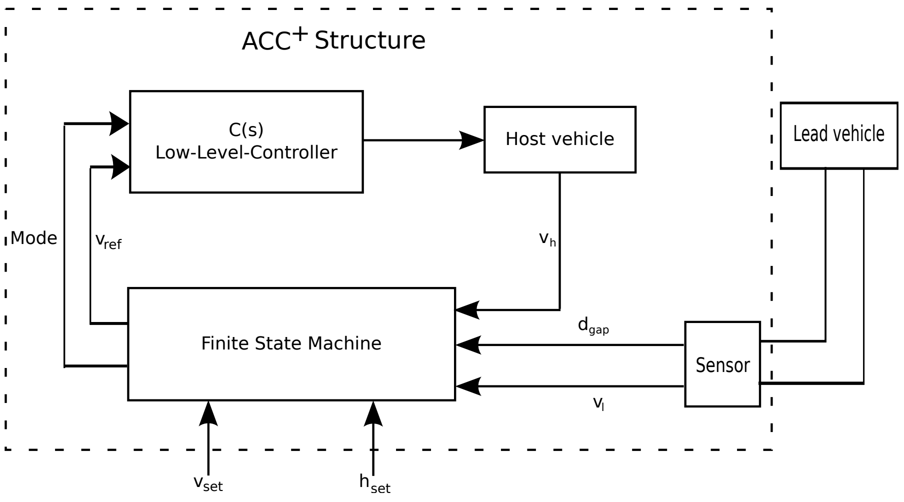


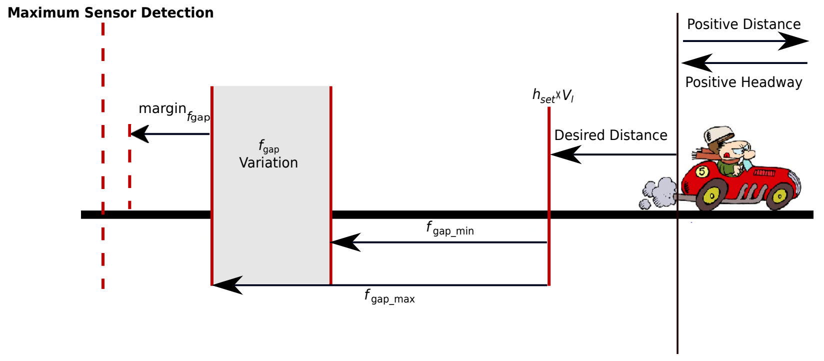
FIGURE 10.2: High level conceptual design block diagram

restriction for the host vehicle's acceleration a_h is the maximum available deceleration B available by applying full brake force, i.e. $a_h \geq -B$. For a chosen value of a_h in this range, the distance required to reduce the host vehicle's velocity to match the lead vehicle's velocity at the desired following distance $h_{set} \times v_l$ is $f_{gap}(v_l, v_h, a_h)$. The size of $f_{gap}(v_l, v_h, a_h)$ is derived from Newton's formula of motion as follows:

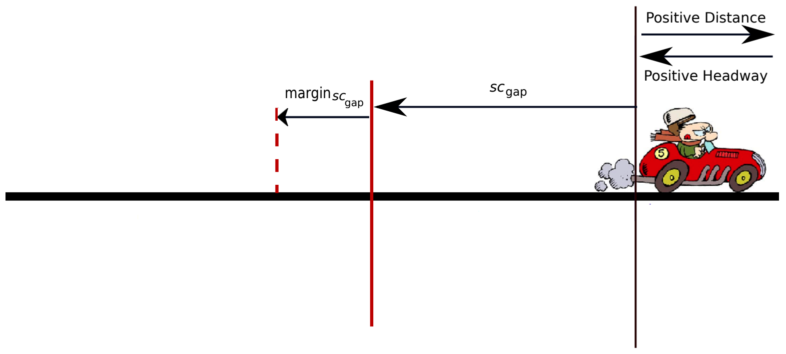
$$f_{gap}(v_l, v_h, a_h) = \frac{v_h^2 - v_l^2}{-2 \times a_h}, \qquad (-B \leq a_h < 0) \tag{10.9}$$

Eq. 10.9 describes the distance required for the host vehicle to achieve v_l as its new velocity, where $-B \leq a_h < 0$ is the deceleration of the host vehicle. According to Eq. 10.9, if the host vehicle wants to use a negative, constant acceleration a_h to achieve the leader's velocity by the time it reaches following distance $h_{set} \times v_l$, it has to start decelerating at distance $f_{gap}(v_l, h_v, a_h)$. Note that once the host vehicle achieves the leader's velocity the size of $f_{gap}(v_l, v_h, a_h)$ will become zero (Eq. 10.9).

Based on the constraints on acceleration and the maximum range of the distance sensor, there are constraints on possible values for f_{gap} to a value between f_{gap_min} and f_{gap_max}. As shown in Fig. 10.3(a), the system bounds on $f_{gap}(v_l, v_h, a_h)$ by restricting the values of a_h that the ACC$^+$ system will use. The upper bound f_{gap_max} and lower bound f_{gap_min} will be defined based upon the upper and lower bound of $a_h < 0$. An upper bound of a_h is the minimum deceleration that the ACC$^+$ system will use by, for example, easing up on the throttle at the current vehicle velocity, while a lower bound is achieved by the maximum braking deceleration B the vehicle can generate based on the current vehicle weight and road conditions (ie. $a_h \geq -B$).



(a) Required distance for approaching the lead vehicle in Follow mode



(b) Minimum required distance in Safety Critical mode

FIGURE 10.3: Required distance in *Follow* and *Safety Critical* mode

To make the system more realistic, another safety margin has been taken into account related to the system delay ϵ that is required to respond to messages from the ACC$^+$ system to the engine and brake controllers and the time they require to activate their respective actuators and have them respond. This margin can be determined by the following formula:

$$margin_{f_{gap}}(v_h, a_h) = \left(\frac{A_{max}}{-a_h} + 1 \right) \left(\frac{A_{max}}{2} \times \epsilon^2 + \epsilon \times v_h \right) \tag{10.10}$$

The size of this margin for the response delay (Eq. 10.10) can be derived in similar fashion to the derivation of $margin_{sc_{gap}}(v_h)$ (Eq. 10.2) by replacing the maximum braking deceleration B with the deceleration a_h. The value of a_h is considered to be negative in all of the formulas given when approaching the lead vehicle, assuming a slower lead vehicle. Once the host vehicle reaches the leader's velocity, it will attempt to track the lead vehicle's velocity and those formulas are not required anymore. Finally, the *Follow* mode will be activated if the relative distance between vehicles, d_{gap}, is less than or equal to $f_{gap}(v_l, v_h, a_h) + margin_{f_{gap}}(v_h, a_h) + (h_{set} \times v_l)$ but greater than the safety critical distance. Note that the value of $f_{gap}(v_l, v_h, a_h)$ becomes negative in the case when the leader's velocity is greater than the host vehicle's velocity ($v_l > v_h$). Therefore, the system always chooses the $\max(f_{gap}(v_l, v_h, a_h), 0)$ for system safety. Although $h_{set} \times v_l$ converges to zero as v_l goes to zero, $margin_{f_{gap}}(v_h, a_h) > 0$ ensures a minimum following distance.

$$d_{gap} \leq max(f_{gap}(v_l, v_h, a_h), 0) + margin_{f_{gap}}(v_h, a_h) + (h_{set} \times v_l) \qquad (10.11)$$

The velocity of the host vehicle should be in a range such that the right side of Eq. 10.11 is within the maximum range of the *Sensor*, as shown in Fig. 10.3(a). Assume the *Sensor* component of the ACC$^+$ system shown in Fig. 10.2 which measures the velocity and position of the lead vehicle relative to the host vehicle has a maximum range of d_{range} meters. Then the maximum v_{set} that can be employed by the system assuming a realistically comfortable deceleration a_h as 30% of maximum deceleration B ($a_h = -0.3B$), time delay ϵ in the response of the system, and a worst case zero velocity of lead vehicle ($v_l = 0$) results in the following equation:

$$v_{set} \leq \sqrt{2 \times 0.3B \times (d_{range} - margin_{f_{gap}}(v_{set}, -0.3B))} \qquad (10.12)$$

Note that Eq. 10.12 represents an approximation of v_{set} because $margin_{f_{gap}}(v_h, a_h)$ has been considered to be a fixed value.

The third mode is the *Safety_ Critical* mode that activates when a vehicle suddenly cuts in the lane or an obstacle appears in front of the vehicle, and the relative distance is less than or equal to the minimum stopping distance for the vehicle when full braking power is applied. In this case the the host vehicle has no choice but to use its maximum braking power to exit the critical zone where $d_{gap} \leq sc_{gap}(v_l, v_h)$ (see Fig. 10.3(b)). Although this situation should not normally occur when the host vehicle is in the follow state, it may happen in critical situations such as a cut-in scenario. Fig. 10.3(b) illustrates $sc_{gap}(v_l, v_h)$. The size of this zone and the margin for the response delay has been derived in Eq. 10.1 and Eq. 10.2 of Section 10.4. According to the discussion in Section 10.4, this mode implies safety and collision-freedom of any ACC$^+$ system. Note that in this scenario, we assume the same maximum braking deceleration for both the host and the lead vehicles.

10.5.2 Mode Switching

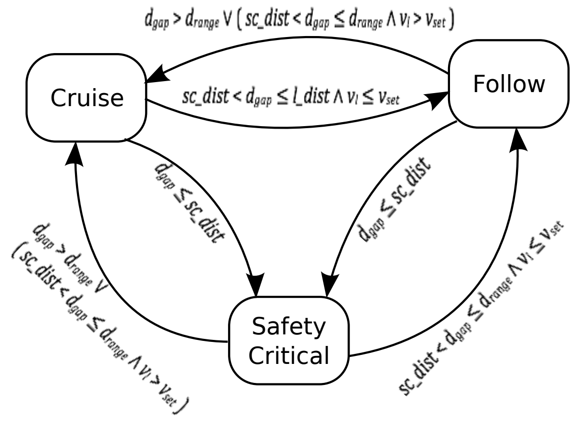
It may, in fact, be the case that the lead and host vehicles have different values for the maximum brake deceleration; hence, the equation for the distance $sc_{gap}(v_l, v_h)$ (Eq. 10.1) may be changed accordingly. Let us assume that the maximum negative acceleration due to maximum brake force for host and lead vehicles are B and b, respectively, then the value of $sc_{gap}(v_l, v_h)$ becomes:

$$sc_{gap}(v_l, v_h) = \frac{v_h^2}{2 \times B} - \frac{v_l^2}{2 \times b} \qquad (10.13)$$

The procedure for deriving this version of sc_{gap} is trivial. The ACC$^+$ system switches to *Safety_ Critical* mode when the relative distance becomes less than or equal to the value of $sc_{gap}(v_l, v_h)$ as formalized in Eq. 10.14.

$$d_{gap} \le sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h) \tag{10.14}$$

According to Eq. 10.1, the safety critical gap shown in Fig.10.3(b) converges to zero when the host vehicle attains the same velocity as lead vehicle (ie. $sc_{gap}(v_l, v_h) \to 0$). This is under the assumption that both vehicles have the same maximum braking deceleration. In the case when the maximum braking deceleration differs (Eq. 10.13), $sc_{gap}(v_l, v_h) > 0$ when $v_h = v_l$ and $B < b$, providing the extra required braking margin due to the lesser maximum deceleration of the host vehicle. In the case when $B > b$, $sc_{gap}(v_l, v_h) < 0$ when $v_h = v_l$ so we take the maximum of $sc_{gap}(v_l, v_h)$ and 0.


$d_{gap} > d_{range} \vee (sc\_dist < d_{gap} \le d_{range} \wedge v_l > v_{set})$
Cruise
$sc\_dist < d_{gap} \le l\_dist \wedge v_l \le v_{set}$
Follow
$d_{gap} \le sc\_dist$
$d_{gap} \le sc\_dist$
$d_{gap} > d_{range} \vee (sc\_dist < d_{gap} \le d_{range} \wedge v_l > v_{set})$
Safety Critical
$sc\_dist < d_{gap} \le d_{range} \wedge v_l \le v_{set}$

where

$$l\_\ dist := max(f_{gap}(v_l, v_h, a_h), 0) + margin_{f_{gap}}(v_h, a_h) + (h_{set} \times v_l)$$
$$sc\_\ dist := max(sc_{gap}(v_l, v_h), 0) + margin_{sc_{gap}}(v_h)$$

FIGURE 10.4: Finite State Machine

Fig. 10.4 and Table 10.3 are alternative representations of the high level design of our ACC$^+$ system. Fig. 10.4 shows the finite state machine (FSM) with the three major modes as separate states. Guard conditions are attached to transitions in this figure. The tabular representation of transition from one state to another is given in Table 10.3. Here the notation $Mode_{-1}$ denotes the previous mode of the FSM.

The actual value of deceleration applied in the *Follow* state could be chosen to be comfortable for the user while achieving a high level of fuel economy, traffic flow, and safety. This deceleration can be described as minimizing a_h. The point that the vehicle switches to the *Follow* mode, can be determined by an optimization process selecting a_h and then the desired velocity reference

Let
$l\_dist := max(f_{gap}(v_l, v_h, a_h), 0) + margin_{f_{gap}}(v_h, a_h) + (h_{set} \times v_l)$
$sc\_dist := max(sc_{gap}(v_l, v_h), 0) + margin_{sc_{gap}}(v_h)$ in

					Mode
$d_{gap} \leq d_{range}$	$d_{gap} \leq sc\_dist$				*Safety_Critical*
$d_{gap} \leq d_{range}$	$d_{gap} > sc\_dist$	$v_l > v_{set}$			*Cruise*
$d_{gap} \leq d_{range}$	$d_{gap} > sc\_dist$	$v_l \leq v_{set}$	$d_{gap} \leq l\_dist$		*Follow*
$d_{gap} \leq d_{range}$	$d_{gap} > sc\_dist$	$v_l \leq v_{set}$	$d_{gap} > l\_dist$	$Mode_{-1} \neq Cruise$	*Follow*
$d_{gap} \leq d_{range}$	$d_{gap} > sc\_dist$	$v_l \leq v_{set}$	$d_{gap} > l\_dist$	$Mode_{-1} = Cruise$	*Cruise*
$d_{gap} > d_{range}$					*Cruise*

TABLE 10.3: Decision making structure of ACC$^+$

will be computed using Eq. 10.15 and sent from the FSM to the low-level controller.

$$v_{ref} = \sqrt{v_l^2 - 2 \times a_h \times (d_{gap} - v_l \times h_{set})} \tag{10.15}$$

Eq. 10.15 can be derived based on the Eq. 10.11 for the normal following action where $-B \leq a_h < 0$. This velocity reference signal is defined for the case that host vehicle detects a slower lead vehicle and the ACC$^+$ system needs to decrease the velocity such that the host vehicle can achieve leader's velocity by the desired headway. The term under the square root in Eq. 10.15 will not be negative as long as the velocities are greater than or equal to zero. In Eq. 10.15 $d_{gap} - v_l \times h_{set}$ represents the distance it takes for the host vehicle to achieve the leader's velocity. If this term is less than zero $(v_l^2 - 2 \times a_h \times (d_{gap} - v_l \times h_{set}) < 0)$, it means that the host vehicle should move backward which is in contradiction with the first assumption of the ACC$^+$ system. Therefore, the system always picks a maximum value between this term $(v_l^2 - 2 \times a_h \times (d_{gap} - v_l \times h_{set}))$ and zero (i.e. $max(v_l^2 - 2 \times a_h \times (d_{gap} - v_l \times h_{set}), 0)$).

In a cut-in scenarios when the velocity of the lead vehicle (v_l) and relative distance (d_{gap}) change abruptly from one set of specific values to another, there is no continuous trajectory for the parameters of the system. It is possible that the *Mode* of the system was *Follow* before the change and remains in *Follow* after updating the sensor values for the new leader. For example, the host vehicle is decreasing the velocity of the host in the *Follow* mode to achieve a leader's velocity by the desired headway when suddenly another leader vehicle l_{new} with velocity less than v_{set}, i.e. $v_{l_{new}} < v_{set}$, cuts in the lane. However, the host vehicle's velocity is less than this new leader's velocity ($v_h < v_{l_{new}}$) and the relative distance between the host vehicle and new leader is not less than safety critical distance (i.e. $d_{gap_{new}} > max(sc_{gap}(v_l, v_h), 0) + margin_{sc_{gap}}(v_h)$). In this example, the ACC$^+$ system will not change the mode of operation and will remain in *Follow*. Therefore, the ACC$^+$ system may accelerate to match the new leader's velocity by the desired headway. After updating different parameters such as v_l, and d_{gap}, the range of a_h for this purpose can be between 0 to A_{max}. Finally, Eq. 10.15 can be used as v_{ref} with

$0 \leq a_h \leq A_{max}$. The optimization process should find a valid value for a_h from this range based on the relative distance, the distance it takes the host vehicle to achieve the leader's velocity, and the desired headway. However, it should be bounded so that the system will not thrash between *Follow* and *Safety_ Critical*. This is a subset of the behaviour we are formally verifying. Although, test cases do not reveal any thrashing in this scenario, further analysis and formal verification are necessary to ensure that the system is free of thrashing. This formal analysis is left to future work that could be done using techniques such as those of [28].

In the *Safety_ Critical* state the desired velocity v_{ref} is set to zero by the FSM. The goal is to continuously apply the maximum brake force to get the host vehicle out of this critical zone. Therefore, the maximum brake command is passed to the low-level controller. In this case, the reference signal will be zero velocity with the *Safety_ Critical* mode signal from the FSM to the Low-Level Controller being interpreted as "apply the maximum brake power and close the throttle".

Mode	v_{ref}
Cruise	v_{set}
Follow	$\sqrt{max(V_l^2 - 2 \times a_h \times (d_{gap} - v_l \times h_{set}), 0)}$
Safety_ Critical	0

TABLE 10.4: Velocity reference signal with respect to the state

Typically it is important in hybrid systems design to avoid rapid mode switching. In the ACC$^+$ system as designed, mode switching between states could cause rapid changes in acceleration which is not comfortable for passengers. Thus in Table 10.3 we avoid rapid mode switching by using hysteresis. When $v_l \leq v_{set}$ and $d_{gap} > l\_dist$, the table checks the previous value of the FSM state, denoted $Mode_{-1}$. The system remains in *Cruise* if the previous value of *Mode* is *Cruise* ($Mode_{-1} = Cruise$), otherwise ($Mode_{-1} \neq Cruise$) it remains in *Follow* or switches from *Safety_ Critical* to *Follow*. This behaviour is similarly defined in the finite state machine Fig. 10.4. Once the current state becomes *Follow* or *Safety_ critical*, the FSM will switch to *Cruise* only in the case that the leader is traveling faster than v_{set} or in the absence of a lead vehicle or object ($d_{gap} > d_{range} \vee (sc\_dist < d_{gap} \leq d_{range} \wedge v_l > v_{set})$). Consequently, the FSM changes state from *Cruise* to *Follow* only in the case that $d_{gap} \leq l\_dist$. The only other potential source of mode thrashing is between *Follow* and *Safety_ Critical*. According to Table 10.4, the reference velocity signal in *Follow* mode is defined as in Eq. 10.15. This v_{ref} ensures that in a typical following of a slower leader, the FSM does not switch back and forth between *Follow* and *Safety_ Critical.* In the case that a lead vehicle cuts in the lane, once the host vehicle exits the critical distance, the ACC$^+$ system switches from *Safety_ Critical* to *Follow* through the guard condition $sc\_dist < d_{gap} \leq d_{range} \wedge v_l \leq v_{set}$ (Fig. 10.4) and system does not fall back in to *Safety_ Critical* due to the definition of v_{ref} in *Follow* mode (Eq. 10.15). Finally, we can provide v_{ref} for the continuous controller based upon the *Mode* of operation (Fig. 10.2). Table 10.4 defines the value of v_{ref} for each *Mode.*

The desired objective in this mode switching is to avoid sudden application of full brake with the resulting severe jerk in non-critical scenarios. The system should not switch to *Safety_ Critical* mode unless a leader vehicle cuts in the lane and there is not enough distance between the host and lead vehicle. There is a particular circumstance under which the mentioned desired objective may be violated during the "normal" functioning of our ACC$^+$ system. This scenario happens when the host vehicle is traveling with reference velocity v_{set} and the ACC$^+$ system's current *Mode* is *Cruise.* If there is a slower lead vehicle in the lane (i.e., $v_l < v_{set}$), but the ACC$^+$ system has not yet changed its *Mode* to *Follow*, we expect the system to switch from *Cruise* to *Follow* when $d_{gap} \leq l\_dist$. However, if the host vehicle's driver decides to change the value of v_{set} to a new value that is less than v_l (i.e., $v_{set_{new}} < v_l$), then according to Fig. 10.4 and Table 10.3, the ACC$^+$ system will not switch the *Mode* from *Cruise* to *Follow* after $d_{gap} \leq l\_dist$. Therefore, the ACC$^+$ system will try to maintain the new desired velocity $v_{set_{new}}$ in *Cruise* mode. In this situation, for a fixed acceleration $a_h < 0$, the required distance for the host vehicle to slow to $v_{set_{new}}$ can be obtained from the following formula:

$$dist_v = \frac{v_h^2 - v_{set_{new}}^2}{-2 \times a_h}, \qquad (-B \leq a_h < 0) \tag{10.16}$$

If this required distance is greater than or equal to the difference between d_{gap} and the safety critical distance $sc_{gap}(v_l, v_h)$, the system will eventually transition directly from *Mode Cruise* to *Safety_ Critical.* Therefore, some additional functionality should be defined in *Cruise* to avoid this undesired behaviour. The system should restrict the driver to choosing the set point velocity from a range of values which do not lead to a full brake in *Safety_ Critical* mode. This range of values can be derived from the above explanation, and is formulated in Eq. 10.17.

$$d_{gap} - sc_{gap}(v_l, v_h) > dist_v \tag{10.17}$$

Note that in Eq. 10.17, $margin_{sc_{gap}}(v_h)$ is not considered for simplicity. A lower bound for $v_{set_{new}}$ can be calculated by replacing $sc_{gap}(v_l, v_h)$ and $dist_v$ in Eq. 10.17 with their formulas. Eq. 10.18 demonstrates the lower bound of v_{set} in *Cruise* mode when the velocity of the leader vehicle (v_l) is lower than the initial set point velocity v_{set} and the driver decides to change v_{set} to a value lower than v_l.

$$v_{set_{new}} > \sqrt{v_h^2(\frac{B-a_h}{B}) + 2a_h(d_{gap} + \frac{v_l^2}{2B})} \tag{10.18}$$

This lower bound for v_{set} in Eq. 10.18 is only for avoiding *Safety_ Critical* mode in normal operation of the ACC$^+$ system. Although the continuous controller in *Cruise* mode manipulates the throttle to decrease or increase the velocity, some percentage of brake can be added to the control action in *Cruise* mode. Therefore, a_h can be picked, for instance, as 10% of the maximum deceleration achieved by full brake B ($a_h = -0.1B$). Eq. 10.19 provides the lower bound for v_{set} by considering 10% of B.

$$v_{set_{new}} > \sqrt{1.1v_h^2 - 0.1v_l^2 - 0.2 \times B \times d_{gap}} \tag{10.19}$$

Note that, if the term under the square root in Eq. 10.18 or Eq. 10.19 becomes negative, it means that $v_{set_{new}}$ can be any value greater than zero ($v_{set_{new}} \geq 0$). Therefore, the lower bound, derived in Eq. 10.19, can be defined by the maximum function in Eq. 10.20.

$$v_{set_{new}} > \sqrt{max(1.1v_h^2 - 0.1v_l^2 - 0.2 \times B \times d_{gap}, 0)} \tag{10.20}$$

As a conclusion, *Cruise* mode operation should be refined based on the following conditions:

No leader / Faster leader: v_{set} can be defined in the interval from zero to the upper limit in Eq. 10.12.

Slower leader: v_{set} can be defined in the interval from the lower limit in Eq. 10.20 to the upper limit in Eq. 10.12.

The implementation of this conditioning operation in *Cruise* mode has been left for future work.

10.5.3 Continuous Controller Design

Other than the case when we are in *Safety_ Critical* mode, the Low-Level (continuous) controller can implement a standard continuous feedback controller designed to meet tracking and disturbance rejection performance requirements. In mode *Cruise* we can use a simple Single Input Single Output (SISO) controller to try to have v_h track v_{ref}. In the case when we are in mode *Follow*, we have to use a Multiple Input Multiple Output (MIMO) controller in order to have v_h track the v_l at a distance of d_{gap}. A typical performance specification of a control system is "good tracking" of the reference signal(s). This is usually interpreted as asymptotic tracking of a single step or ramp reference signal and is commonly met with a standard design such as a PID controller. However, PID controllers typically do not maintain reasonable performance in the presence of uncertainties in the plant model and set of reference signals. Therefore, robust controller design techniques have been developed to achieve performance in terms of a weighted norm bound that result in strong performance in the presence of plant uncertainties such as weight of the loaded vehicle, friction of the road, *etc.* Among all possible structured and unstructured uncertainties, we choose simple disk-like multiplicative uncertainty to simplify our analysis. We then design our Low-Level Controller based on the Loopshaping analysis technique of [7]. As a result, our ACC$^+$ system attains reliable performance in the presence of plant uncertainty for a variety of reference signals (Table 10.4).

10.5.4 Simulation Results

In this section, a test case is presented in Fig. 10.5 to evaluate the behaviour of the proposed ACC$^+$ system design on a scale model vehicle. Although all the possible scenarios cannot be captured with one test case, we try to capture the most significant behaviour to examine the performance of our ACC$^+$ system.

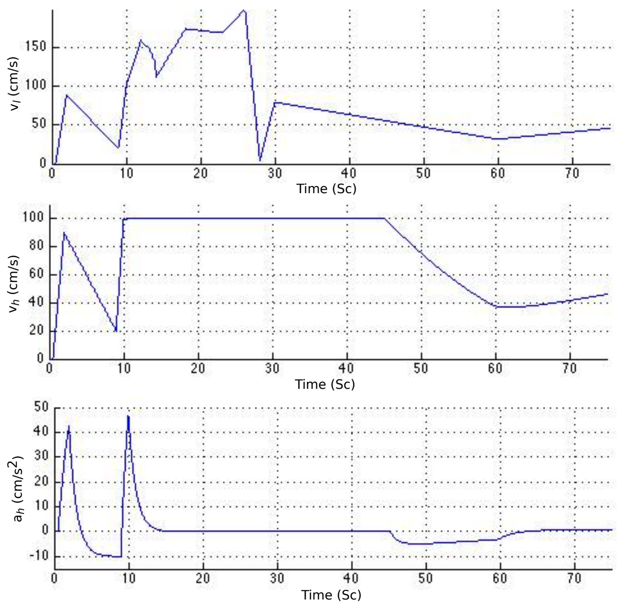


FIGURE 10.5: Simulation Results

Fig. 10.5 illustrates the behaviour of the host vehicle controlled by our ACC$^+$ system in various conditions such as when a lead vehicle is present at varying velocities or absent. The first plot in Fig. 10.5 depicts the velocity behaviour of the leader vehicle (v_l) which is measured in centimetre per second (cm/s). According to this plot, the leader starts from an initial velocity of zero ($v_l = 0$) and changes its velocity as shown. The host vehicle starts at a certain desired headway at the beginning of the simulation. Therefore, the host vehicle tracks the leader's velocity in *Follow* mode until the leader travels faster than the host vehicle's set point velocity (v_{set}) at time $t = 10$. This behaviour can be seen in the second plot of Fig. 10.5, where the host vehicle's velocity is shown. The host vehicle's set point velocity is defined as 100 cm/s ($v_{set} = 100$). Therefore, the second plot of Fig. 10.5 shows the host vehicle tracks the leader's velocity until it exceeds $v_{set} = 100$ cm/s and the ACC$^+$ mode of operation is changed from *Follow* to *Cruise*. As shown in the first plot of Fig. 10.5, the leader vehicle decreases its velocity to lower values than the host's v_{set} at approximately $t = 27$. However, the ACC$^+$ system does not immediately change its mode of operation and the host vehicle continues with $v_h = 100$ cm/s in *Cruise* mode as long as the relative distance between the two vehicles is greater than the required distance for following the leader. The ACC$^+$ system changes the mode from *Cruise* to *Follow* only when d_{gap} becomes less than or equal to *l_ dist*. This occurs around $t = 45$ when the host vehicle decreases its velocity in order to attain the leader's velocity by the desired headway. Consequently, the host vehicle matches its velocity with that of the lead vehicle while maintaining the desired headway.

The third plot of Fig. 10.5 shows the host vehicle's acceleration. According to this plot, the acceleration increases when the host vehicle is accelerating its velocity at first. The acceleration increases from zero to approximately 40 cm/s^2 at first while the host vehicle's velocity increases from zero to 90 cm/s. The acceleration does not become negative right after the host vehicle starts decreasing its velocity from 90 cm/s to 20 cm/s. The reason for the host vehicle still having a positive acceleration, even when the lead vehicle is decelerating, is the presence of an integral term in the continuous controller. When initially developing the safety verification of the refined ACC$^+$ controller that appears in the following section, the specification stated that if $v_l < v_h$ in *Follow* mode, then the acceleration of the host vehicle chosen by the controller must satisfy $a_h < 0$. Clearly a reasonable linear control system design, such as the one simulated in Fig. 10.5, does not satisfy this property. The verification in the following subsection was modified to allow positive accelerations even in the case when $v_l < v_h$ precisely for this reason. The lesson here is that one has to be careful to make sure that the formal model that is verified faithfully models the actual system.

The host vehicle changes its acceleration in order to maintain a required velocity. However, due to the nature of the continuous controller, the acceleration control it generates does not change instantaneously due to the continuous dynamic of the system. This ACC$^+$ system attains the leader's velocity by a desired headway if the leader travels slower than the host vehicle's set point velocity. In addition, this system will continue to track the leader' velocity after the desired headway is achieved. In the absence of a slower leader, the system's objective is to track a desired set point velocity. As shown in Fig. 10.5, our ACC$^+$ system behaves safely and will not switch to *Safety_ Critical* mode in this particular normal operation scenario. This control structure is not conservative because the required safety constraints are considered as a separate mode of operation and do not affect the operation of *Cruise* and/or *Follow* modes. Therefore, the required safety constraints and the desired performance could be obtained simultaneously by this design in this scenario.

10.5.5 Verification

In this section we provide a formalization for the refined mode switching of the ACC$^+$ system described in Section 10.5.2 using *differential dynamic logic* (d$\mathcal{L}$). This formalization is presented in **Model 2**. We use the dynamic operations of the host and lead vehicles as defined in Section 10.4 (Eq. 10.4 & Eq. 10.5). The leader vehicle behaviour is the same as **Model 1** in Section 10.4, where acceleration can be chosen from the valid range (Eq. 10.6) (Line(3)). The non-deterministic repetition $*$ and parallel operation of the host and leader vehicle has been already defined in **Model 1** (Line (1-2)). The *Other* functionality of the ACC$^+$ system in **Model 1** is now formalized as successive actions to capture the other driving modes. The host vehicle controller takes action in a more restricted manner. Safety constraints must be satisfied related to relative distance, velocities and the selected velocity for cruise mode. The host vehicle has three different operating modes that are represented sequentially in (4).

Three operating modes *Cruise*, *Follow*, and *Safety_ Critical* always use the current value of $sc_{gap}(v_l, v_h)$, $margin_{sc_{gap}}(v_h)$, and d_{gap} in Line (5) to randomly choose the desired acceleration non-deterministically within the valid range to control the speed of the host vehicle under given safety margins.

The *Cruise* operating mode states that if d_{gap} is not less than or equal to $sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$ and the speed of the lead vehicle is greater than v_{set} then either the acceleration of the host vehicle a_h can be assigned non-deterministically from $-B \leq a_h \leq A_{max}$ when speed of the host vehicle v_h is less than or equal to v_{set}, or a_h can be assigned non-deterministically from $-B \leq a_h \leq A_{max}$ when the speed of the host vehicle v_h is greater than v_{set}. This operating mode is formalized in (6), where the speed of the host vehicle always maintains according to the selected driver speed considering the speed of lead vehicle and safety margins. The range of deceleration is formalized based on the band limit of the continuous controller. For instance, if the continuous controller has an integral term, the acceleration might continue for some time with a value greater than zero even when $v_h > v_{set}$; hence, when performing the verification the range of a_h cannot be restricted to a value between $-B$ to 0 in the case when $v_h > v_{set}$. By a similar reasoning, the range of a_h cannot be restricted to a value between 0 to A_{max} in the case that $v_h \leq v_{set}$. Thus, we consider the range of $-B \leq a_h \leq A_{max}$ for both mentioned cases. In the work of [13], the behaviour of the continuous controller is not taken into account. The verified ACC controller in Loos *et al.* [13] may not apply to certain controllers, such as PID controller, since the band limit of the continuous controller is not included in the verification. This concept was explained in detail in Section 10.5.4. Although the host vehicle's acceleration can be formalized based on the continuous controller (i.e. for a PID controller $a_h := k_1 + k_2 \times \int v_h(t).dt + k_3 \times \frac{\mathrm{d}}{\mathrm{d}t} v_h(t)$), we use the possible physical range ($-B \leq a_h \leq A_{max}$) to capture a class of possible continuous controllers. Accordingly, the continuous controller can be an arbitrary design without any concern about the safety properties of ACC$^+$ system.

The *Follow* operating mode is applicable only when the speed of the lead vehicle is less than or equal to the driver selected desired speed, (i.e. $v_l \leq v_{set}$) and the vehicle separation is not in the safety critical zone.. The *Follow* operating mode is specified in statement (7) of **Model 2** that specifies that if d_{gap} is not less than or equal to $sc_{gap}(v_l, v_h) + margin_{sc_{gap}}(v_h)$ and the speed of lead vehicle is less than or equal to v_{set}, then the acceleration of the host vehicle a_h can be assigned non-deterministically based on the current status of the host and lead vehicles' velocities (v_h and v_l). If the host's velocity is greater than the leader's velocity ($v_h > v_l$) then a_h should be negative ($-B \leq a_h < 0$). This case happens normally when the ACC$^+$ system detects a slower leader vehicle and should decrease its current velocity gradually in order to maintain v_l. Although a_h should be negative in this case, we considered its value between $-B$ and A_{max} ($-B \leq a_h \leq A_{max}$) in the formalization. This demonstrates that the continuous controller may work with a positive acceleration for a short time interval until obtaining a negative value. Therefore, $-B \leq a_h \leq A_{max}$ is not in contradiction with the expected behaviour in *Follow* mode by the reasoning provided in Section 10.5.4. After maintaining v_l, the ACC$^+$ system should track the leader's behaviour. Therefore, if the leader accelerates, and $v_h \leq v_l$, then of the possible values in $-B \leq a_h \leq A_{max}$ one would expect the controller to trend towards values of $a_h \geq 0$. As mentioned earlier, a_h cannot be switched from a negative to a positive value instantly due to the continuous behaviour of the system. Therefore, a_h cannot be formalized between 0 to A_{max} in the last case when $v_h \leq v_l$, otherwise significant jerk may occur in the system. As a result, $-B \leq a_h \leq A_{max}$ is a reasonable range to be considered for any arbitrarily continuous controller.

The ACC$^+$ system can make any of these two choices according to the situation. Furthermore, the current values of $f_{gap}(v_l, v_h, a_h)$ and $margin_{f_{gap}}(v_h, a_h)$ are calculated sequentially, where system must satisfy $d_{gap} - (h_{set} \times v_l) \leq f_{gap}(v_l, v_h, a_h) + margin_{f_{gap}}(v_h, a_h)$. The test checks that the host vehicle is within $f_{gap}(v_l, v_h, a_h)$ to make sure that the transition to *Follow* is done properly and there is enough distance to achieve v_l as the new host velocity. If the test condition does not hold ($d_{gap} - (h_{set} \times v_l) > f_{gap}(v_l, v_h, a_h) + margin_{f_{gap}}(v_h, a_h)$), then execution will fail. Therefore, this assertion forces the system operation to maintain enough distance for taking an appropriate action in the *Follow* mode. Although this test dose not have any impact on the proof of safety and collision-freedom of **Model 2**, we have defined this assertion to allow the conformity of this formalization to the actual mode switching system of Section 10.5.2.

In the case that this test cannot be satisfied, there are two possible outcomes. First there is the normal behaviour in the presence of a slower leader when the sensor detects a slower leader, but the host vehicle is still able to travel at $v_h = v_{set}$ until it comes within $f_{gap}(v_l, v_h, a_h)$ of the lead vehicle.

The second case for this violation is the opposite problem where there is not enough of a gap to reduce the host vehicle to v_l by the time the host vehicle is within the desired headway h_{set}. However the $f_{gap}(v_l, v_h, a_h)$ distance is always greater than the minimum stoping distance $sc_{gap}(v_l, v_h)$ in the presence of a slower leader vehicle. Further, the maximum delay for the ACC$^+$ system to react to a change, ϵ, has also been taken into consideration during the calculation to estimate the additional safe distance margin for $f_{gap}(v_l, v_h, a_h)$ in order to provide sufficient time for the controllers to react. Line (7) formalize the behaviour of the *Follow* mode in the case when the ACC$^+$ system starts to decrease the host vehicle's velocity to match a slower leader's velocity by the time the host vehicle reaches distance $h_{set} \times v_l$, and then track the leader's velocity at an appropriate distance as long as the leader does not travel faster than v_{set}. This formalization also captures the behaviour of the *Follow* mode after the host vehicle starts to track the leader's velocity. Line (8) formalizes the *Safety_ Critical* mode as defined previously in **Model 1**. The sampling time and dynamic evolution of the system are defined in Line (9), similar to **Model 1**.

The main purpose of the refined ACC$^+$ formalization given in **Model 2** is to investigate the safety of the ACC$^+$ system in the presence of a leader in front of the host vehicle. Additionally we also want to ensure that the vehicle behaves safely when switching between the different modes of operation. The host vehicle's behaviour when the lead vehicle is out of range of the sensor is the same as conventional cruise control systems. The *Cruise* mode controls the speed of the host vehicle on behalf of the driver. In the current formal model of the refined ACC$^+$ system, d_{range} has not been defined (i.e., we assume that the sensor range is effectively infinite). It is left as a further work to prove the correctness of the system with a limited range sensor under the conditions outlined in (Eq. 10.12).

For now, we consider that there is a leader vehicle in the same lane as the host vehicle in **Model 2**. The system checks whether it can satisfy the safety property in the case when an obstacle or lead vehicle is detected. Once the path is cleared from any obstacle or there is no longer a lead vehicle, then it can switch back to the *Cruise* mode to maintain the desired speed (v_{set}).

The proposed ACC$^+$ design has three operating modes, where the system is switching from one mode to another according to desired situation considering safety constraint. The safe distance formula is the most important invariant that must be always satisfied by the ACC$^+$ system in all the operating modes as stated by the Controllability property (Eq. 10.8) in Section 10.4. We wrote **Model 2** in the KeYmaera theorem prover's input language to further demonstrate that this ACC$^+$ system design is safe and collision free as long as the safety critical distance condition (Eq. 10.8) has not been violated.

$$\text{Controllability Condition (10.8)} \rightarrow [\text{Refined ACC}^+]\;\; x_h < x_l \qquad (10.21)$$

The precondition for the formula 10.21 is similar to that of formula 10.7. It indicates that for all iterations of the Refined ACC$^+$ (**Model 2**), the system is collision free ($x_h < x_l$) if the controllability condition (10.8) is satisfied. This fact confirms that the ACC$^+$ system in **Model 2** is a refinement of **Model 1**. Therefore any system, such as **Model 2**, will be safe as long as the controllability condition (10.8) is maintained. We will further investigate the refinement and refactoring relations between **Model 1** and **Model 2** in the next subsection. The safety of a complex model, such as **Model 2**, can be proved based on an abstract model, such as **Model 1**. The refactoring relation makes the proof procedure easier than the procedure we have done for proving relation 10.21. Additionally, the refinement relation allows designers to add new requirements to a system and/or change some parts of the system without violating the required safety properties. Consequently, it can be shown that **Model 2** is derived from the abstract model of Section 10.4 (**Model 1**) by adding some new states and refining the system's behaviour while preserving the required safety properties.

10.5.6 Safe Refactoring

Direct proof of safety and other properties of a complex cyber physical system is often difficult if not impossible due to the complex interaction between software and hardware models. Different approaches have been investigated to overcome this fact such as over-approximating the reachable set of states and defining an abstract model in order to reduce the complexity [11]. The abstract model then can be verified for safety purposes. However, an important part of this method which is typically disregarded in this area is to prove that the original, complex system model is a property preserving refinement of the proposed abstraction. After verifying safety of an abstract model of a cyber physical system, any update in any part of that model requires reverification of the whole new system. Refinement reasoning makes the reverification process easier by assuring that the new additional part of the system does not violate the safety of the whole system. Platzer and his coworkers recently proposed a refinement relation for systems described in differential dynamic logic (d$\mathcal{L}$) in [14]. Mitsch *et. al* in [14] introduced two notions of refinement "*Projective Relational Refinement*" and "*Partial Projective Relational Refinement*". According to [14]:

> "***Projective Relational Refinement:*** *Let $V \subseteq \Sigma$ be a set of variables. Let $|_V$ denote the projection of relations or states to the variables in V. We say that hybrid program α refines hybrid program γ w.r.t the variables in V ($\alpha \sqsubseteq^V \gamma$) iff $\rho(\alpha)|_V \subseteq \rho(\gamma)|_V$.*"

where ρ is the transition relation used to specify reachable states.

Although we used KeYmaera [24] to prove safety property (Eq. 10.21) of **Model 2** in Section 10.5.5, we want to further investigate refinement reasoning. We defined a safe abstract model of any ACC or ACC$^+$ system in

Section 10.4 and proved the collision-freedom property of that model. In this section we want to show that **Model 2** refines **Model 1**. The Projective Relational Refinement definition holds for **Model 2** with respect to **Model 1** since the reachable states of **Model 2** are a subset of the reachable states of **Model 1**. We can thus conclude that **Model 2** refines **Model 1** with respect to the variables of these models (**Model 2** $\sqsubseteq^V$ **Model 1**). Therefore, **Model 2** inherits the collision freedon safety property from **Model 1**. We will use refactoring methods from [14] to demonstrate the validity of this claim.

Mitsch et. al [14] developed "proof-aware refactoring" and proposed some rules with associated proof obligations to define a refinement relation in terms of refactoring. Two refactoring *Structural* and *Behavioral* are defined in [14]. "*Structural refactoring changes the structure of a hybrid program without changing its reachable states*"; while, Behavioral refactoring partially changes the reachable states. Therefore, some auxiliary proof obligations are necessary to demonstrate inheritance of safety or correctness properties in behavioral refactoring. We use "safety relational refinement" and "auxiliary safety proof" from [14] for refinement reasoning.

> "*Safety relational refinement. Prove that all reachable states from the refactored model α are already reachable in the original model γ.*"

> "*Auxiliary safety proof. Prove that a refactored model α satisfies some safety properties under the assumption of an existing proof about the original model γ. The auxiliary safety proof patches this proof w.r.t. the changes made by the refactoring. Let $\forall^\gamma$ quantify universally over all variables that are changed in γ. The intuition is that, assuming $\models \forall^\gamma(\phi \rightarrow [\gamma]\phi)$ (ϕ is an inductive invariant of γ), we can close the identical parts in the proof from the assumption by axiom and only need to show correctness for the remaining, new parts of the refactored model. For auxiliary safety use an invariant of $\mathcal{I}(\phi) \equiv (\phi \wedge \forall^\gamma(\phi \rightarrow [\gamma]\phi))$ for the refactored program α to prove $(F \wedge \mathcal{I}(\phi)) \rightarrow [\alpha^*]\psi$.*"

where F is some formula based on the definition of partial projective relational refinement. A hybrid program α is a partial refinement of γ with respect to some variables in the set of variables V and some formula F ($\alpha \sqsubseteq_F^V \gamma$) if and only if $(?F; \alpha) \sqsubseteq_F^V (?F; \gamma)$. In the case that $F \equiv true$, this partial refinement relation becomes a total refinement relation ($\alpha \sqsubseteq^V \gamma$ iff $\alpha \sqsubseteq_{true}^V \gamma$). Therefore, F in $(F \wedge \mathcal{I}(\phi)) \rightarrow [\alpha^*]\psi$ is an additional condition for partial refinement cases.

According to the above definitions from [14], we want to show that if abstract model, **Model 1**, guarantees a safety property, i.e. collision-freedom, then this safety property can be proven for a refactored model, **Model 2**. We translate our problem using the auxiliary safety proof method as shown below:

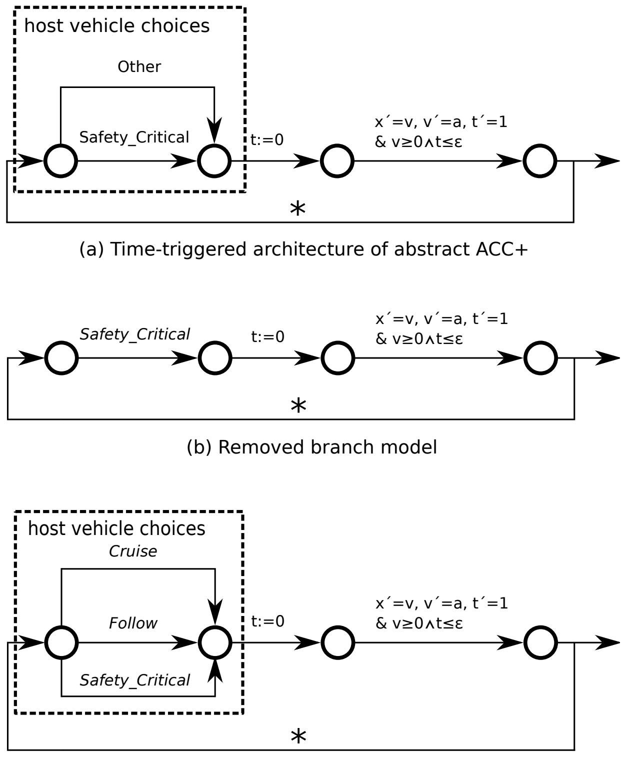


FIGURE 10.6: Time-Triggered Architecture of ACC$^+$

α is "Refined ACC$^+$" (**Model 2**)

γ is "Abstract ACC$^+$" (**Model 1**)

ϕ is "Condition (10.8)"

ψ is $x_h < x_l$

We already proved that: ($\phi \to [\mathit{Abstract\ ACC^+}]\ x_h < x_l$) as Eq. 10.7. We want to show the same collision-freedom ($x_h < x_l$) is valid for refactored model, in this case the refined ACC$^+$ ($\phi \to [\mathit{refined\ ACC^+}]\ x_h < x_l$). Therefore, we should strengthen the inductive invariant of **Model 1**, Controllability Condition (10.8), with the safety approved assumption for abstract model.

$$\mathcal{I}(\phi) \equiv (\phi \wedge \forall x \forall v(\phi \to [\text{Abstract ACC}^+]\ \phi))$$

We want to formally prove that: $\mathcal{I}(\phi) \to [\text{Refined ACC}^+]\ x_h < x_l$

The *Event- to time-triggered architecture* refactoring changes a hybrid program from event-triggered to timed triggered. This refactoring process separates the continuous evolution of the system from control choices. Fig. 10.6(a) shows the time-triggered architecture of **Model 1** (Abstract ACC$^+$) as a state transition system. The procedure of deriving this architecture can be found in [14]. Fig. 10.6(b) demonstrates removing one branch (*Other*) from the original model (Fig. 10.6(a)) while the safety property is still preserved and then Fig. 10.6(c) introduces two new branches (*Cruise* and *Follow*) to Fig. 10.6(b) without changing the *Safety_ Critical* branch. Both figures (Fig. 10.6(b) & Fig. 10.6(c)) depict the "*Introduce Control Path*" refactoring, which is defined under the category of "*Behavioral Refactorings*" in [14]. Finally, Fig. 10.6(c) shows the time-triggered architecture of **Model 2** (Refined ACC$^+$). The safety proof procedure of this refactored model can then be constructed from Fig. 10.6(b) and Fig. 10.6(c). The details of this proof are left as future work. Although this procedure provides easier steps in the safety proof of the refined system, we want to further demonstrate that one transition is split into two transitions. The two new transitions cover the same guard condition while each transition has a subset of the old transition's behaviour. Therefore, we want to further prove the case splitting of the *Other* (old transition), which means that *Cruise* and *Follow*, as new transitions, result in a subset of the behaviour of *Other*.

Another use of refinement of these two models (i.e. **Model1** and **Model2**) is to demonstrate that the abstract model can be improved and adapted to a more complex model in order to meet new requirements. We want to use a refactoring from [14] to prove a safety property of a refined model based on the abstract one. However, proof-aware refactoring in [14] does not propose any path-split (case splitting) refactoring. In other words, we need to add an additional refactoring proof in *differential dynamic logic* (d$\mathcal{L}$) to show that a transition in the abstract model can be split in to two or more new branches. In this case the *Other* transition is split without touching the *Safety_ Critical* case in order to preserve safety of the whole new system.

We want to show that the *Other* mode in **Model1**, Fig.10.1, is split to two new modes *Cruise* and *Follow* in **Model2**, Fig. 10.4, without touching the *Safety_ Critical* mode. This fact can be established by proving that new branches apply in the same situations as the old branches and each will not violate the acceptable range for parameters of the old branch. In our example, a_h in *Cruise* and *Follow* will not be out of the acceptable range which has been already defined in *Other* ($-B \leq a_h \leq A_{max}$). Therefore, another notion of refactoring and refinement (path-split) can be introduced in the proof refactoring of *differential dynamic logic* (d$\mathcal{L}$) that can be show to preserve safety properties.

Consequently, the whole refactoring procedure with path-split refinement can be done more easily than the steps which we have done based on "*proof-aware refactoring*". Using this technique we can deduce Fig. 10.6(c) from Fig. 10.6(a) directly, without the intermediate step shown in Fig. 10.6(b). Providing the formal syntax and semantics of path-split refinement is again left as future work.

10.6 CONCLUSION

CC and ACC systems have been used by several car companies to regulate the speed of the car in restricted traffic situation with a required minimum speed. These systems are not suitable for a very low speed, traffic jam environment. ACC$^+$ extends CC and ACC features to provide automatic regulate speed of car in these types of traffic environment. In this work we have verifed an abstract ACC$^+$ system behaviour guarantees that the system is collision free and safe under all possible scenario other than when a car cuts in front of the host vehicle inside the safety critical stopping distance. ACC and CC systems have already been formalized to verify their correct behaviour and safe operation, however formal methods have not previously been applied to ACC$^+$ systems to verify the system requirements and desired system behaviour. We have presented formal verification of a more realistic hybrid control system for ACC$^+$ systems using dynamic logic (d$\mathcal{L}$) and proposed a new path-split refinement in the process. Future work includes the implementation of the ACC$^+$ system on a hardware platform to validate that formal model faithfully captues the system behaviour, requirements and safety properties.

Model 2: Formalization of refined ACC^{+} system

$$
\begin{array}{rcll}
\mathit{ACC}^{+} & \equiv & (\mathit{Vehicle}; \mathit{Drive})^{*} & (1)\\[1em]
\mathit{Vehicle} & \equiv & \mathit{host} \mid\mid \mathit{leader}; & (2)\\[1em]
\mathit{leader} & \equiv & a_l = *; ?(-B \leq a_l \leq A_{max}) & (3)\\[1em]
\mathit{host} & \equiv & \mathit{Calc\_sc_{gap}};\ \mathit{Cruise};\ \mathit{Follow};\ \mathit{Safety\_Critical}; & (4)\\[1em]
\mathit{Calc\_sc_{gap}} & \equiv & sc_{gap}(v_l, v_h) := \frac{v_h^2 - v_l^2}{2 \times B}; & \\
 & & \mathit{margin}_{sc_{gap}}(v_h) := (\frac{A_{max}}{B} + 1)(\frac{A_{max}}{2} \times \epsilon^2 + \epsilon \times v_h); & \\
 & & d_{gap} := x_l - x_h; & (5)\\[1em]
\mathit{Cruise} & \equiv & \text{if } \big(\neg(d_{gap} \leq sc_{gap}(v_l, v_h) + \mathit{margin}_{sc_{gap}}(v_h)) \wedge v_l > v_{set}\big) \text{ then} & \\
 & & \big(?(v_h \leq v_{set}); a_h := *; ?(-B \leq a_h \leq A_{max})\big) \bigcup & \\
 & & \big(?(v_h > v_{set}); a_h := *; ?(-B \leq a_h \leq A_{max})\big) & \\
 & & \text{fi}; & (6)\\[1em]
\mathit{Follow} & \equiv & \text{if } \big(\neg(d_{gap} \leq sc_{gap}(v_l, v_h) + \mathit{margin}_{sc_{gap}}(v_h)) \wedge v_l \leq v_{set}\big) \text{ then} & \\
 & & \big(?(v_h > v_l); a_h := *; ?(-B \leq a_h \leq A_{max})\big) \bigcup & \\
 & & \big(?(v_h \leq v_l); a_h := *; ?(-B \leq a_h \leq A_{max})\big) & \\
 & & f_{gap}(v_l, v_h, a_h) := \frac{v_h^2 - v_l^2}{-2 \times a_h}; & \\
 & & \mathit{margin}_{f_{gap}}(v_h, a_h) := (\frac{A_{max}}{-a_h} + 1)(\frac{A_{max}}{2} \times \epsilon^2 + \epsilon \times v_h); & \\
 & & ?(d_{gap} - (h_{set} \times v_l) \leq f_{gap}(v_l, v_h, a_h) + \mathit{margin}_{f_{gap}}(v_h, a_h)) & \\
 & & \text{fi}; & (7)\\[1em]
\mathit{Safety\_Critical} & \equiv & \text{if } \big(d_{gap} \leq sc_{gap}(v_l, v_h) + \mathit{margin}_{sc_{gap}}(v_h)\big) \text{ then} & \\
 & & a_h := -B & \\
 & & \text{fi}; & (8)\\[1em]
\mathit{Drive} & \equiv & t := 0; (x_h' = v_h \wedge v_h' = a_h \wedge x_l' = v_l \wedge & \\
 & & v_l' = a_l \wedge t' = 1 \wedge v_h \geq 0 \wedge v_l \geq 0 \wedge t \leq \epsilon) & (9)
\end{array}
$$

Bibliography

[1] Erika Abrahám-Mumm, Ulrich Hannemann, and Martin Steffen. Verification of hybrid systems: Formalization and proof rules in PVS. In *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on*, pages 48–57. IEEE, 2001.

[2] Nikos Aréchiga, Sarah M Loos, André Platzer, and Bruce H Krogh. Using theorem provers to guarantee closed-loop system properties. In *American Control Conference (ACC), 2012*, pages 3573–3580. IEEE, 2012.

[3] H. Arioui, S. Hima, and L. Nehaoua. 2 DOF Low Cost Platform for Driving Simulator: Modeling and Control. In *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, pages 1206–1211, 2009.

[4] J. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. *Software Engineering Journal*, 8(4):189–209, Jul 1993.

[5] Gabriel Ciobanu and Stefan Rusu. Verifying adaptive cruise control by π-calculus and mobility workbench. Technical Report FML-08-01, Institute of Computer Science Iaşi, December 2008.

[6] Werner Damm, Carsten Ihlemann, and Viorica Sofronie-Stokkermans. Decidability and complexity for the verification of safety properties of reasonable linear hybrid automata. In *Proceedings of the 14th international conference on Hybrid systems: Computation and Control*, pages 73–82. ACM, 2011.

[7] John Comstock Doyle, Bruce A Francis, and Allen Tannenbaum. *Feedback control theory*, volume 1. Macmillan Publishing Company New York, 1992.

[8] O.J. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen. Development of a driver information and warning system with vehicle hardware-in-the-loop simulations. *Mechatronics*, 19(7):1091 – 1104, 2009. Special Issue on Hardware-in-the-loop simulation.

[9] B.A. Guvenc and E. Kural. Adaptive cruise control simulator: A low-cost, multiple-driver-in-the-loop simulator. *Control Systems, IEEE*, 26(3):42–55, 2006.

[10] S. Jairam, K. Lata, S.K. Roy, and N. Bhat. Verification of a MEMS based adaptive cruise control system using simulation and semi-formal approaches. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pages 910–913, 2008.

[11] Taylor T Johnson, Jeremy Green, Sayan Mitra, Rachel Dudley, and Richard Scott Erwin. Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In *FM 2012: Formal Methods*, pages 252–266. Springer, 2012.

[12] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *Proceedings of the 17th International Conference on Formal Methods*, FM'11, pages 42–56, Berlin, Heidelberg, 2011. Springer-Verlag.

[13] S.M. Loos, D. Witmer, P. Steenkiste, and A. Platzer. Efficiency analysis of formally verified adaptive cruise controllers. In *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pages 1565–1570, Oct 2013.

[14] Stefan Mitsch, Jan-David Quesel, and André Platzer. Refactoring, refinement, and reasoning. In *FM 2014: Formal Methods*, pages 481–496. Springer, 2014.

[15] Rasul Mohammadi. *Fault diagnosis of hybrid systems with applications to gas turbine engines.* PhD thesis, Concordia University, 2009.

[16] José Eugenio Naranjo, Carlos González, Ricardo García, and Teresa De Pedro. ACC$^+$ Stop & Go maneuvers with throttle and brake fuzzy control. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):213–225, 2006.

[17] L. Nehaoua, H. Mohellebi, A. Amouri, H. Arioui, S. Espie, and A. Kheddar. Design and control of a small-clearance driving simulator. *Vehicular Technology, IEEE Transactions on*, 57(2):736–746, 2008.

[18] David L Parnas, A John van Schouwen, and Shu Po Kwan. Evaluation of safety-critical software. *Communications of the ACM*, 33(6):636–648, 1990.

[19] George A Peters and Barbara J Peters. *Automotive Vehicle Safety.* CRC Press, 2003.

[20] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 216–232. Springer, 2007.

[21] André Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

[22] André Platzer. *Logical Analysis of Hybrid Systems - Proving Theorems for Complex Dynamics.* Springer, 2010.

[23] André Platzer. The complete proof theory of hybrid systems. In *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, pages 541–550. IEEE, 2012.

[24] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems (system description). In *Automated Reasoning*, pages 171–178. Springer, 2008.

[25] M.E. Russell, A. Crain, A. Curran, R.A. Campbell, C.A. Drubin, and W.F. Miccioli. Millimeter-wave radar sensor for automotive intelligent cruise control (icc). *Microwave Theory and Techniques, IEEE Transactions on*, 45(12):2444–2453, Dec 1997.

[26] P. Shakouri and A. Ordys. Application of the state-dependent nonlinear model predictive control in adaptive cruise control system. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 686 –691, oct. 2011.

[27] Olaf Stursberg, Ansgar Fehnker, Zhi Han, and Bruce H Krogh. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice*, 12(10):1269–1278, 2004.

[28] Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach.* Springer, 2009.

[29] D.J. Verburg, A.C.M. Van der Knaap, and J. Ploeg. Vehil: Developing and testing intelligent vehicles. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 537–544 vol.2, 2002.

[30] Y. Yamamura, M. Tabe, M. Kanehira, and T. Murakami. Development of an adaptive cruise control system with stop-and-go capability. Technical Report 2001-01-0798, SAE, 2001.

[31] E Zaloshnja, T Miller, F Council, and B Persaud. Comprehensive and human capital crash costs by maximum police-reported injury severity within selected crash types. In *Annual proceedings / Association for the Advancement of Automotive Medicine. Association for the Advancement of Automotive Medicine*, volume 48, pages 251–263, 2004.