

Formal Methods for Safety-Critical Embedded Real Time System: A case Study

R. Gurulingesh[†]

03329008

KReSIT, IIT Bombay

[†] *guru@it.iitb.ac.in*

Abstract

There is a great interest in ensuring correctness of safety-critical embedded real time systems. This paper presents a formal model based on timed automata for Adaptive Cruise Control system, a safety critical component in Automotive Electronics. This is done by correctly abstracting the system and splitting the continuous variables of environment into finite set of regions. Properties that this timed model need to satisfy are specified which can be verified using some existing tools.

Keywords:

Timed automata, Safety-critical embedded real time system, Adaptive Cruise Control, Sensors.

1. Introduction

Formal specifications have been a focus of software engineering research for many years and have been applied in wide variety of applications successfully. Formal methods provide:

- More precise specification
- Ability to verify designs before executing them during test

Their industrial usage has considerably increased over the years to large class of applications, safety-critical systems being one of the main targets. Its of utmost importance to ensure correctness of safety-critical embedded real time systems since on one hand the use of software gives greatly increased functionality and flexibility and on the other hand it provides unprecedented possibilities for errors.

There exist a number of modeling and verification tools for real-time systems; for instance, tools based on the theory of timed automata such as KRONOS and UPPAAL.

Correct abstraction of large systems will result in verifiable models from which one can always infer properties of original model. This is illustrated in the paper by considering Adaptive Cruise Control (henceforth abbreviated as ACC) as a case study. ACC is a safety-critical embedded real time system that interacts with the environment via a controller and sensor. ACC properties are parameterized with free variables, which should be determined in order to prove the safety of the system. Theory of timed automata is used in this paper to model the system and abstract out the system to build a model for ACC by dividing the continuous environment into finite set of regions.

2. Adaptive Cruise Control system

Adaptive Cruise control system is a “driver assistance” system designed to enhance the safety of driver. ACC builds on functionalities of the existing cruise control system and within certain limits, maintains appropriate distance from the vehicles in front and road speed at all times automatically. ACC maintains a constant safe distance to the vehicle in front, automatically slowing down if the vehicle in front slows down, and automatically speeding up if the vehicle in front picks up speed. It operates in conjunction with existing

throttle control, braking, vehicle detection and other subsystems. Figure 1 shows the block diagram of the system.

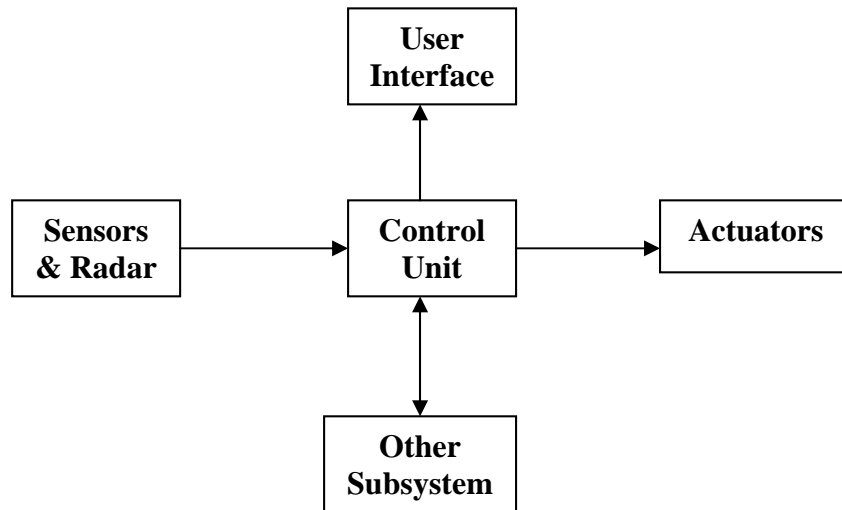


Figure 1. Block diagram of ACC showing the interaction with the components

ACC monitors host vehicle's speed, status of cruise control, driver's intervention, data from vehicle detection system. Using all data ACC does some intelligent computation to achieve the goal of keeping safe distance of separation by using throttle or brake. Actions taken by ACC are reflected immediately in the User Interface to keep driver informed which is very crucial in critical situations. It also interacts with other subsystems like Anti-lock Braking system, lane-control system, and road signal communicating system to adapt certain parameters depending on the situation which are beyond the scope of this paper and will not be detailed here.

To develop a verifiable model, paper keeps the system to very basic functionalities ignoring certain advanced features listed above which can be added later. ACC system is abstracted out and it can be viewed as shown in Figure 2.

The simplified system consists of:

1. Environment
2. Sensor and Radar subsystem
3. Control Unit

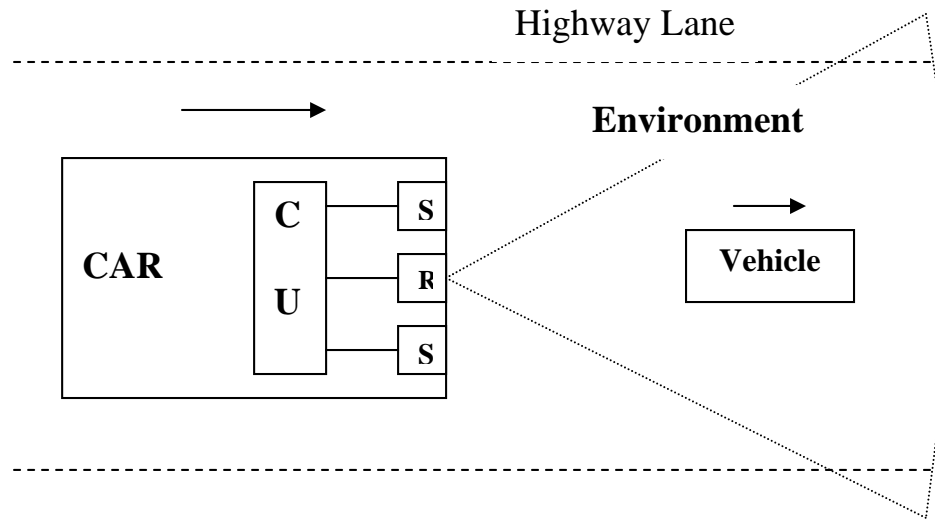


Figure 2. Block diagram of abstraction of ACC with environment

2.1 Environment: It is a dynamic system consisting of several static objects and moving vehicles in front of host car. Relative velocity and distance of leading vehicles with respect to host are of interest to the system. Note that these values are in continuous domain and determine the behavior of ACC. Certain assumptions are being made about the environment which will be stated later to develop the verifiable model following the “Keep it Simple” policy.

2.2 Sensors and Radar subsystem: This subsystem provide the necessary interface through which ACC can keep up-to-date information about the environment behavior and other necessary information that is critical for the system to decide the course of action. Sensors and Radar consists of components for sending and receiving signals and a signal processing unit to interpret the data received to get useful information which will be used by Control Unit. These can operate in periodic (dumb sensors) or event-driven (intelligent sensor) manner. They provide the information such as relative velocity and distance of leading vehicle, current velocity of host car, brake and throttle position, etc.

2.3 Control Unit (CU): This is the main component of the system which maintains up-to-date information about the environment with the help of sensors and radar and does some intelligent computation to take the necessary actions to achieve the goal of system. It will have collection of tasks and will run on a embedded computer with Real time

operating system like RTLinux, OSEK, etc. The tasks will have information about environment behavior, interacts with other subsystems, and decides actions to be taken, and instructs actuators to perform those actions.

3. Formal Modeling

In this section, ACC system is modeled using timed automata, and certain properties that system should satisfy are listed which can be verified using existing tools like KRONOS, UPPAAL, etc.

3.1 ACC Environment

As described earlier, environment is a dynamic system consisting of several vehicles moving with a different velocity at different distance from the host vehicle. To make system simple and to develop a verifiable model, environment can be restricted to the lane in which host vehicle is moving. Vehicle i in the environment moves with velocity v_i at a distance d_i . Longitudinal distance of the leading vehicle w.r.t. host vehicle or lane can also be ignored without much loss of the information. Leading vehicle's relative velocity and leading distance are of importance to the system. Relative velocity is given by:

$$v_{rel} = v_i - v_{host}$$

where v_i is the velocity of i^{th} leading vehicle

v_{host} is the velocity of host vehicle

Its reasonable enough to assume that radar only keeps track of information related to the nearest vehicle to the host car.

3.1.1 Regions:

As area in the environment is continuous, it can be divided into different regions which will help the control unit to take the actions depending on location of the leading vehicle. The area is classified into six regions as shown in Figure 3. Numbers have been assigned to each region and Table 1. gives this numbering scheme. Note that region "Away" is not of our concern as radar cannot detect the vehicles in that region. ACC never allows a leading vehicle to enter the Danger-zone area; it warns the driver well in advance that leading

vehicle may enter this zone and goes to idle state where it expects driver to handle the situation manually. These two regions are listed for the sake of completeness.

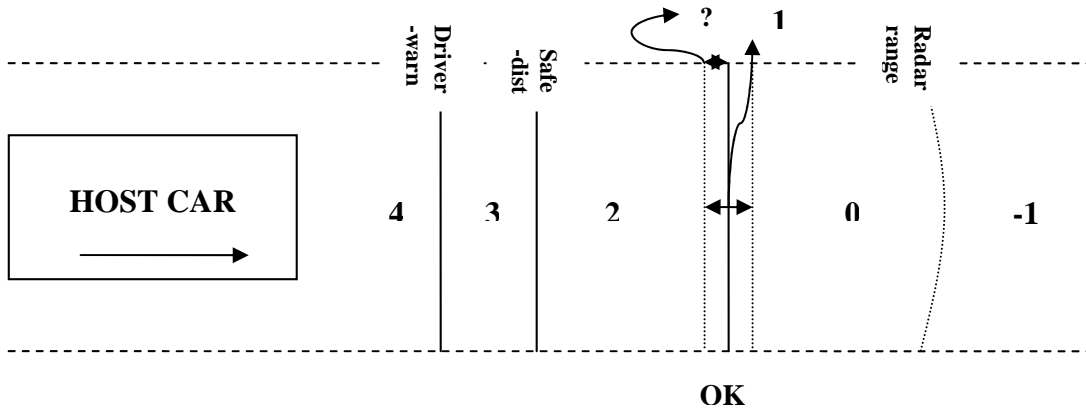


Figure 3. Splitting the environment into different regions

3.1.2 Assumptions:

The assumptions made about environment listed here will help to simplify the system and to develop a verifiable model:

1. Bound the minimum and maximum relative velocity of leading vehicle
2. Only the nearest vehicle in the environment is tracked by radar
3. No vehicle will suddenly cut-in the lane under consideration.
4. Leading vehicle will not be allowed to enter danger-zone region, a warning will be issued to driver well in advance and the system disengages which driver has to reset after the situation is under control.
5. Leading vehicle will cover the regions either in ascending or descending order. No abrupt change in the locality of vehicle under observation is permitted.

<u>Region No.</u>	<u>Region Name</u>
-1	Away
0	Far
1	OK
2	Close
3	Too-Close
4	Danger-zone

Table 1. Listing of environment region number with naming

3.1.3 Timed Automata of Environment:

Figure 4 shows timed automata of the environment which is based on the regions listed in table 1 and the time constants in the figure can be calculated by attaching certain values to relative velocity bounds.

Initially, environment will be in “*Idle*” state or location where no vehicles are in front of the host car. Later as vehicle comes within radar’s range it moves to “*Far*” location. Details of all possible transitions between states are listed in table 2.

<u>Transition</u>	<u>Condition for Transition</u>	<u>Set on Transition</u>
1	$d \leq \text{radar-range}$ and $d > \text{OK+?}$	$x = 0$
2	$d > \text{radar-range}$, $d = -1$	$x = 0$
Stay in ‘Far’	$x \leq \text{max-time-bound-far}$	
3	$x \geq \text{min-time-bound-far}$, $d = \text{OK+?}$	$x = 0$
4	$d > \text{OK+?}$ and $d < \text{radar-range}$	$x = 0$
Stay in ‘OK’	$x \leq \text{max-time-bound-OK}$	
5	$x \geq \text{min-time-bound-OK}$, $d = \text{safe-dist}$	$x = 0$
6	$d > \text{safe-dist}$ and $d < \text{OK-?}$	$x = 0$
Stay in ‘Close’	$x \leq \text{max-time-bound-Close}$	
7	$x \geq \text{min-time-bound-Close}$, $d = \text{warn-dist}$	$x = 0$
8	$d > \text{warn-dist}$ and $d < \text{safe-dist}$	$x = 0$
Stay in ‘TooClose’	$x \leq \text{max-time-bound-TooClose}$	
9		

Table 2. List of all possible transitions with labels on transitions for timed automata of the environment. ‘x’ in the table denotes the clock.

The bounds like min-time-bound(s), max-time-bound(s) for all the locations/states can be calculated using relative velocity bounds and the length of respective regions. The action

taken by the CU depends on the location in which the leading vehicle is present and whether it is approaching or separating. The goal of CU will be to maintain a safe distance of separation i.e. maintain “*lock-on*” distance and follow the vehicle as explained later. CU can also check for changing lane when the leading vehicle is in “*Close*” region and distance of separation is reducing between both vehicles. It can also interact with the roadside signals and ABS to adapt parameters; features like these are not considered.

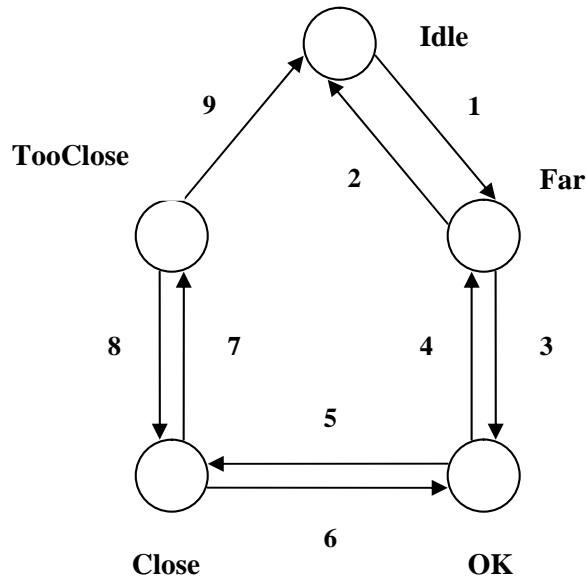


Figure 4. Timed automata of environment of ACC system

3.2 Sensors and Radar

Different kinds of sensors are used in this system for different purposes. Wheel sensors are needed to compute the correct velocity of host car. Sensors near brake and throttle control are needed to determine the current position of the respective controllers. Radar sensor scans the environment for the objects and gives relative velocity and distance of nearest vehicle in front of the car.

Sensors can operate in either periodic or event-driven manner. In this paper they are assumed to be of periodic activities. Figure 5 shows the timed automata of wheel sensor and radar.

Initially, Sensor is in “*Idle*” state and waits for its period. Once its period comes, it goes to “*Busy*” state and computes the speed of the host car according to its perspective and

updates the value $v w_i(t)$ and then set clock y to 0 where t is the time when i^{th} wheel sensor scans the velocity. This activity repeats periodically.

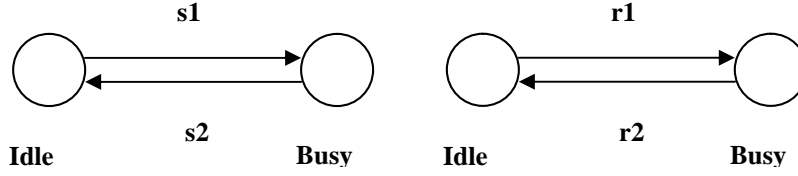


Figure 5. (a) Timed automata of sensor (b) Timed automata of radar

On the similar line, radar initially in “Idle” state goes to “Busy” state on its period and calculates relative velocity and distance of leading vehicle and sets the values $rv(t)$ and $rd(t)$ and sets clock z to 0 where rv is relative velocity and rd is relative distance. The periods of all these sensors are assumed to be harmonic which will help in simplifying CU functionalities.

ACC system usually has four wheel sensors placed at four wheels of car to compute the velocity and radar will be positioned in the front portion of car.

3.3 Control Unit

It is collection of tasks running on a real time operating system platform on a single processor. It executes several tasks such as keeping up-to-date information about the environment, deciding course of action depending on the sensors and radar information and instructing actuators to carry out these actions. This paper groups all the tasks into two without affecting the system much. These tasks are:

1. Merging Sensors
2. Information Base and Decision Making

which are detailed below.

3.3.1 Merging Sensors:

Here, CU receives individual values from all wheel sensors and computes a single value for host car velocity which it will use in further decision making step. One of the simplest ways to combining these values is taking average of all readings.

In the i^{th} step:

$$curr-vel(i) = \sum v w_j(i) / 4$$

where $j = 1 \dots 4$

It also reads radar values i.e. relative velocity and distance of leading vehicle and stores in shared variables for access by other tasks:

$$rel-vel(i) = rv(i), dist(i) = rd(i)$$

Figure 6 shows the automata for this periodic task doing the above mentioned computations every period. Note that the 'period' in this timed automaton should be harmonic to sensor and radar periods or making all the periods equal will simplify the things.

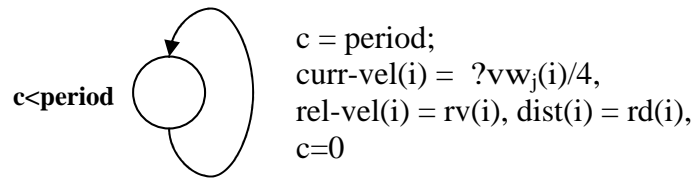


Figure 6. Timed automata of Sensor Merging with 'c' as clock variable

3.3.2 Information Base and Decision Making:

It is a part of CU that's makes use of data from "Merging Sensor" part to maintain accurate information about the environment and take necessary action after analyzing the situation carefully.

Figure 7 shows the timed automata for this part of CU. Initially, system is in "Idle" state. If any vehicle comes in contact with radar, it will make a transition to "Far" state. It will periodically read the values current velocity of host car, relative velocity and distance of leading vehicle. Also, depending on whether distance of separation is increasing or decreasing, CU will send accelerate by $i \text{ m/s}^2$ i.e. $a(i)$ or decelerate by $i \text{ m/s}^2$ i.e. $d(i)$ command to actuators. These values are computed by CU so as to achieve the lock-on distance with the leading vehicle and follow it eventually. Note that acceleration and deceleration values are usually bounded to 3 m/s^2 for ACC. So, a simple control algorithm would be to choose the value whichever makes distance of separation close to lock-on distance.

All the possible transitions and actions are listed in Table2.

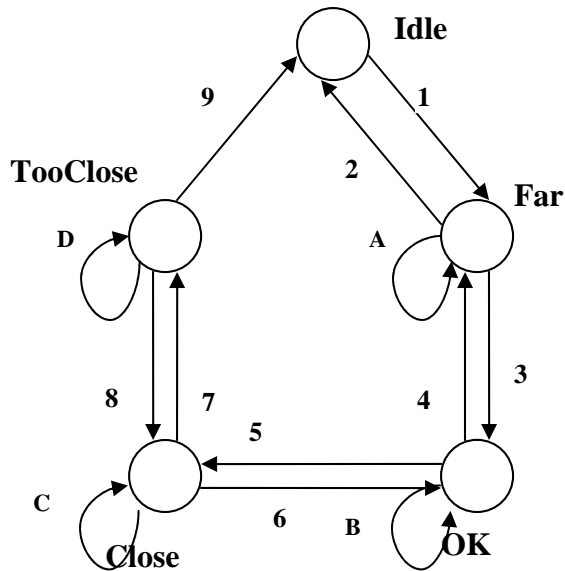


Figure 7. Timed automata of Information Base and Decision Making task

<u>Transition</u>	<u>Condition for Transition</u>	<u>Set Values</u>	<u>Action</u>
1	dist <= radar-range and > OK+?	p=0, q=0	
2	dist > radar-range, q=p1	q=0	
Stay in 'Far'	q<p1, p<p2	_____	
3	dist <= OK+? and > OK-?, q=p1	q = 0	
4	dist > OK+?, q=p1	q=0	
Stay in 'OK'	q<p1, p<p2	_____	
5	dist <= OK-? and > safe-dist, q=p1	q=0	
6	dist > OK-?, q=p1	q=0	
Stay in 'Close'	q < p1, p < p2	_____	
7	dist <= safe-dist and > warn-dist, q=p1	q=0	
8	dist > safe-dist, q=p1	q=0	
Stay in 'TooClose'	q<p1, p<p2	_____	
9	q>min-time-bound-warn	q=0, p=0	Warn Driver
A	p=p2, d<=radar-range and > OK+?	p=0	d(i) or a(i)
B	p = p2, d <= OK+? and > OK-?	p=0	d(i) or a(i)

C	$p = p2, d \leq \text{OK- ? and } > \text{ safe-dist}$	$p=0$	$d(i) \text{ or } a(i)$
D	$q < \text{min-time-bound-warn}, p = p2, d \leq \text{safe-dist and } > \text{ warn-dist}$		$d(i) \text{ or } a(i)$

Table 2. All possible transitions in timed automata of CU where p and q are clock variables and p1 is the period of radar and p2 is the period of CU’s decision making task.

4. Verification

The primary goal of ACC is to maintain the safe distance between host and leading vehicle by maintaining accurate information about the environment and satisfying some of the liveness, safety and timeliness properties. Some of liveness and safety properties to be satisfied and accuracy of information maintained by CU about environment is measured in terms of time. Timeliness plays a crucial role in specifying and verifying properties for real time systems. Some of the properties are stated below using UPPAAL syntax. UPPAAL uses simplified version of Computation Tree Logic for specifying the properties of the system. In the properties listed below, ‘A[] p’ means for all paths p always holds. They can be verified using existing timed automata modeling tools.

1. Information Base of CU has an accurate information about the vehicle in the environment. The information maintained by Information Base (IB) of CU about the distance of vehicle ‘*IB.dist*’ should not deviate too much from the environment’s information about the same. This can be modeled as:

$A[] (IB.dist - dist \leq P)$ where P is the maximal difference between these two values.

2. Leading vehicle will never come in ‘danger-zone’.

$A[] (IB.dist > driver-warn-dist)$

3. CU does not issue false alarms. For instance, it will never report advancement of leading vehicle toward the host car before the vehicle actually does so.

$A[] (IB.dist \leq dist)$

4. System will not enter *deadlock* state.

$A[] (\text{not } deadlock)$

5. Conclusions and Future Work

Adaptive Cruise Control is an interesting safety-critical real time system to study which is a mixture of both discrete and continuous subsystems. Continuous subsystem being the environment here which is discretized to finite set of regions in order to develop a verifiable model simplistically. This paper provided a case study of ACC describing it formally using timed automata and listed some of the properties that it needs to satisfy which can be verified using existing tools.

The paper made many assumptions which are too restrictive and can be relaxed to make the study more interesting and challenging. Assumptions like radar keeping track of only nearest vehicle, no cut-in situation, restricting to single lane, not considering driver's manual intervention if relaxed result in a challenging and complex system. For instance, radar can have one more state where it keeps track of other vehicles in the environment. So, relaxing the assumptions and enhancing functionalities of the system and making use of tools like KRONOS and UPPAAL to verify the properties of system will result in an interesting future work.

5. References

1. R. Alur and D. Dill. *A Theory for Timed Automata*, Theoretical Computer Science, volume 125, p. 183-235, 1994
2. Ioannou, P. and Chien, C.C. *Autonomous Intelligent Cruise Control*, IEEE Transaction on Vehicle Technology, Vol.42, No. 4, p. 657-672, 1993.
3. Mayr, R. and Bauer, O. *Safety issues in intelligent cruise control*, in proceedings of IEEE international conference on Intelligent Transportation Systems, p. 970-975, Oct. 5-8, 1999.
4. Alexandre David. *Uppaal2k: Small Tutorial*, on <http://www.uppaal.com/>