Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Adaptive Cruise Control with Timed Automata

Mustafa Yavuz Kara * **Ebru Aydin Gol** *

* *Department of Computer Engineering, Middle East Technical University, Turkey (e-mail: myavuzkara@gmail.com, ebrugol@metu.edu.tr).*

Abstract: An adaptive cruise control (ACC) system maintains the vehicle at the given target speed when there is no leading vehicle in the sensor range. On the other hand, in the presence of a leading vehicle, the system maintains a safe distance between the vehicles while driving as close as possible to the target speed. For such an automated system, besides meeting safety requirements, it is also important to provide a comfortable drive. In this paper, we develop a formal model for adaptive cruise control system based on timed automata and express specifications in temporal logics. The proposed model supports different acceleration levels. Parametric constraints govern the transitions to the states associated with acceleration levels. The proposed parameter optimization methods generate parameter valuations for particular driving styles while guaranteeing safety and the specifications over the target speed. Therefore, the resulting system is guaranteed to satisfy the requirements while the driver comfort is optimized. The models and the synthesis approach are illustrated with examples.

Keywords: Timed automata, adaptive cruise control, parameter synthesis.

1. INTRODUCTION

A cruise control system is a driver assistance system that aims at making the driving more comfortable and convenient for the driver. The only functionality provided by a conventional cruise control system is keeping the vehicle at a preset target speed, which is determined by the driver. An adaptive cruise control (ACC) system extends the conventional cruise control systems with the ability to follow a leading vehicle in the same lane at a safe distance. When there is no leading vehicle within the sensory range, the ACC system maintains the vehicle at the target speed (SAE J2399, 2014). However, when there is a leading vehicle within the sensor range, ACC system adjusts the speed by accelerating or decelerating (negative acceleration) and follows the leading vehicle at a safe distance unless the leading vehicle accelerates beyond the target speed, in which case the system only accelerates up to the target speed. Two cars driving on the same lane illustrating the ACC problem is shown in Fig. 1.

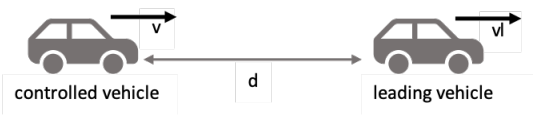

Fig. 1. Two vehicles with speed v, vl and distance d.

While the cruise control systems are considered driver assistance systems, which implies that the system's operation will be supervised by the human driver at all times, safety is still of paramount importance. Therefore, the correctness of such automated systems should be guaranteed. This paper presents a formal methods approach to generate ACC systems based on timed automata.

Timed automata (TA) is a formalism for modeling real-time systems. A timed automaton can simply be viewed as a finite automaton extended with a set of real-time variables, called clocks, which capture the time (Alur, 2015). The clocks enrich the semantics and the constraints on the clocks restrict the behavior of the automaton. An established approach to verifying correctness of TA models is model checking, where the specifications are expressed in a formal language and efficient algorithms are used to check if the model satisfies the given specification. In a TA, the clock constraints are defined as inequalities, e.g., $x < 5$ for clock x. A TA is called parametric when parameters are used in place of constants, e.g., $x < p$. For a parametric TA, the goal of the parameter synthesis problem is to find the set of all parameter valuations such that when the parameters are replaced with the corresponding valuations the resulting TA satisfies the given specification. In general, the parameter synthesis problem is undecidable (Beneš et al., 2015; Étienne, 2019). However, the integer valued parameter synthesis problem from a bounded search space is decidable (Beneš et al., 2015). In addition, efficient parameter synthesis methods based on monotonicity properties exist for specific classes of TA such as L/U automata (Hune et al., 2002).

In this paper, the ACC system, the leading vehicle and the distance between the vehicles are modeled as timed automata (TA). For the ACC system, the automaton states represent the acceleration levels. Parametric constraints are defined to govern the transitions between the accelera-

⋆ This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 798482.

tion modes. The leading vehicle has a similar model but it changes its state (acceleration level), thus its speed, non-deterministically. Finally, the distance automaton synchronizes these TAs and models the distance between the vehicles with respect to their speeds. The system specifications including the safety requirements are expressed formally in Computation Tree Logic (CTL). For example, the safety requirement is formulated as $\forall\Box(d > d_{min})$, i.e., the distance should be greater than the minimum safety distance d_{min} at all times. The specification over the target speed is defined as $\exists \diamond (v = v_{target})$, i.e., the target speed can be reached.

The ACC design problem is studied extensively in control and traffic communities (Jing Zhou and Huei Peng, 2005; Nilsson et al., 2016; Xiao and Belta, 2019; Magdici and Althoff, 2017). In earlier works, the control architecture and low level controller design are studied (Jing Zhou and Huei Peng, 2005). Considering the critical nature of the problem, recent works focus on designing correct by construction systems. Nilsson et al. (2016) designs ACC system from formal specifications via fixed point computations. Whereas, Xiao and Belta (2019) employs control barrier functions for ACC design. In these works, the vehicle is modeled as a lumped point mass and the force to be applied is computed. As in this paper, Larsen et al. (2015) presents a TA model for ACC design, where the controller output is the acceleration level. The TA from (Larsen et al., 2015) has a single acceleration and a single deceleration level. The transitions are non-deterministic and a feedback control automaton is synthesized to guarantee safety. Whereas, in this work, a parametric TA is generated for any given number of acceleration levels. Then the parameters of the TA are synthesized in an automated way such that the resulting TA is guaranteed to satisfy the specifications while the driver comfort is optimized. Furthermore, even though the parametric TA does not belong to L/U class, it is shown that the parameters are monotonic and this property is used to synthesize optimal parameters in an efficient way.

2. ADAPTIVE CRUISE CONTROL PROBLEM

2.1 System Description

A vehicle is modeled with discrete time motion dynamics:

$$v_{k+1} = v_k + a_k, a_k \in \mathbb{U} \tag{1}$$

where, at time step k, v_k is the speed and a_k is the acceleration (or deceleration for $a_k < 0$) that takes values from a finite set $\mathbb{U}$ with $0 \in \mathbb{U}$. The speed limits are denoted by v_{min} and v_{max}.

The vehicles are equipped with a distance sensor. The sensor can detect the relative position of a leading vehicle when it is in the sensor range, which is denoted by d_{range}. The target speed set by the driver is denoted by $v_{target} \in [v_{min}, v_{max}]$. In the considered setup, the ACC system knows its own speed v_k, receives the measured distance $d \in [0, d_{range}]$ of the leading vehicle if there is one from the sensor system, and then sets the acceleration level a_k. The leading vehicle have the same dynamics (e.g. the same acceleration levels). However, its acceleration, speed and control logic are unknown to the controlled vehicle. There are two cases in which a new vehicle enters to the sensor range. First, a vehicle driving on the same lane enters the range at d_{range}, e.g. a slower vehicle appears in the front. Second, a vehicle can enter via a lane switch at a distance $d \in [d_{lane}, d_{range}]$, where d_{lane} denotes the minimum lane switch distance under normal driving conditions [1]. The second case is only allowed when there is no other leading vehicle to avoid too close manoeuvres.

2.2 Specifications

The first specification is the crash avoidance. A safe distance, d_{min}, between the controlled and the leading vehicles should be maintained, i.e,

$$Safety: \ d > d_{min} \ at \ all \ times$$

Second, the vehicle should reach the target speed when it is safe.

$$TargetSpeed: \ reach \ v_{target} \ when \ safe.$$

Finally, the ACC system should keep the speed within the limits, i.e.,

$$SpeedLimits: \ v_{min} \leq v_{target} \ at \ all \ times$$

For the driver comfort, frequent accelerations and decelerations should be avoided and the vehicle should maintain its speed at the target when it is safe (e.g. no oscillation around v_{target}). In addition, sharp decelerations should be avoided unless it is necessary.

Problem: *Given system* (1)*, design an ACC that satisfies the specifications while optimizing the driver comfort.*

2.3 Example System

The parameters of an example system are defined as follows. The speed limits are $v_{min} = 10m/s$ and $v_{max} = 30m/s$. The acceleration levels are $\mathbb{U} = \{-2, -1, 0, 1\}$. The target speed is $v_{target} = 20m/s$ which can be changed by the driver. The sensor range is $d_{range} = 150m$. The minimum allowed lane change distance is $d_{lane} = 100m$. The safety distance is $15m$ (e.g. half of the maximum distance traveled in a second).

3. TIMED AUTOMATA MODELS FOR ACC

In this section, we describe the developed TA models and their formal specifications for the adaptive cruise control problem [2]. For detailed information on TA and temporal logics, we refer the interested reader to Alur (2015). The overall model consists of three timed automata. The first automaton shown in Fig. 2 models the distance measured by the controlled vehicle and synchronizes the TA network. The second (Fig. 3) and the third (Fig. 4) automata model the speed of the controlled and the lead vehicles, respectively.

[1] Other systems such as the emergency brake system should be active when this assumption is violated.

[2] The TA extensions implemented in UPPAAL such as committed locations and integer variables are used in the models (UPP, 2005).

3.1 The Distance Automaton

The distance automaton can be seen in Fig. 2. It has the initial location *wait* and five committed locations marked with "c". Time can not pass in a committed location. In other words, when it is reached, the next transition must be from there. Thus, in distance TA, time can only pass at location *wait* and the transitions through other locations are all instantaneous.

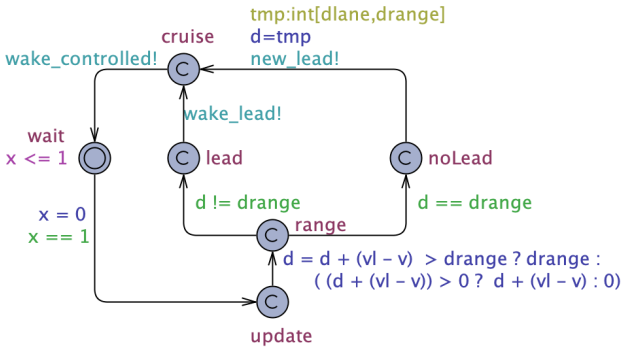

Fig. 2. Distance TA: Synchronizes the network and updates the distance.

The TA has a single clock named x, it is reset on the transition leaving *wait*. *wait* has invariant $x \leq 1$ and x should be 1 to take the transition. Hence, at every second, TA leaves *wait* and makes a loop through either *lead* or *noLead*. First, it goes to *update* when x is 1 and resets the clock. Then, it goes to range and updates the distance between the lead and the controlled vehicles with respect to their speeds v and vl that are modeled by the control TA and the lead TA, respectively. Note that the distance, d, is set to *drange* when the computation returns a higher value to mimic the sensor range. Then, it goes to *lead* or *noLead* with respect to the computed d to model the constraint that a lane switch can only occur when there is no leading vehicle. In the lane switch case, *new_lead* signal is sent to the lead TA and the distance is set to a random value in $[dlane, drange]$ modeling the restrictions from Sec. 2.1. If there is a leading vehicle, *wake_lead* signal is sent to the lead TA. After that, it goes to *cruise* location, and then finalizes the loop by going to the *wait* with *wake_controlled* signal sent to the control TA. Control and lead TAs compute their speeds only when they receive a signal from the distance TA.

3.2 The Cruise Control Automaton

First, the control automaton for the example defined in Sec. 2.3 is described, and then a method to construct a control TA for any number of deceleration levels is defined.

The example system has two deceleration levels and a single acceleration level. The corresponding control TA is shown in Fig. 3. The TA is composed of an initial location called *wait* and four committed locations: *decide* and a location for each level l_0, l_1, and l_2. It does not have a local clock. It only leaves *wait* and goes to *decide* when it receives *wake_controlled* signal from the distance automaton. Then it goes to one of the acceleration locations l_0, l_1, and l_2 or to *wait* based on the measured distance d (computed in distance TA) and its current speed v.

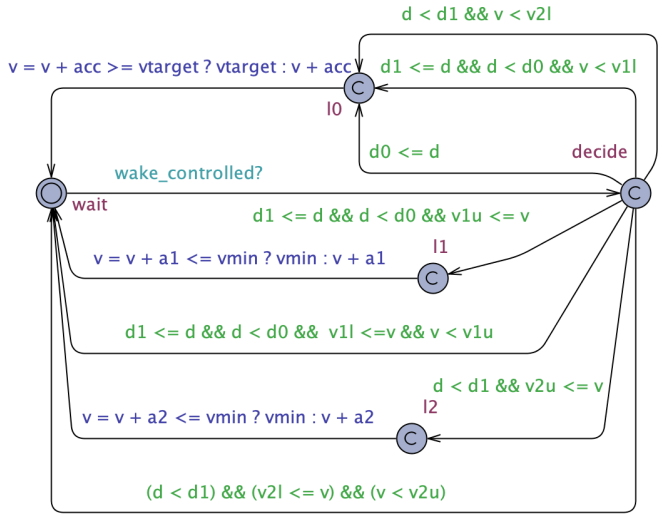

Fig. 3. Control TA: Implements the cruise control schema.

We present the control TA construction method for any number of acceleration levels. Let $\mathbb{U} = \{a_0, 0, a_1 \ldots, a_m\}$ be the acceleration levels given in decreasing order, i.e., $a_i > a_{i+1}$ for $i = 1, \ldots, m-1$ (single positive acceleration level a_0). The control automaton is constructed with $2+m$ locations and $1+4m$ transitions. In particular, in addition to the *wait* and *decide* locations and the wake transition from *wait* to *decide*, 1 new location and 4 new transitions are introduced for each level except $0 \in \mathbb{U}$. Note that the only positive acceleration level is a_0. For a_0, committed location l_0, a transition from *decide* to l_0 with guard $d_0 \leq d$, and a transition from l_0 to *wait* with the following update rule are added to the model:

$$v = \begin{cases} v + a_0 \text{ if } v + a_0 \leq v_{target} \\ v_{target} \text{ otherwise} \end{cases}$$

For each $a_i \in \mathbb{U} \setminus \{0, a_0\}$, the following steps are performed to construct the parametric TA:

- add committed location l_i
- add a transition from *decide* to l_i with guard (slow down with a_i)
$$d_i \leq d \;\&\&\; d < d_{i-1} \;\&\&\; v_i^u \leq v$$
- add a transition from *decide* to *wait* with guard (keep the speed constant at v)
$$d_i \leq d \;\&\&\; d < d_{i-1} \;\&\&\; v_i^l \leq v \;\&\&\; v < v_i^u$$
- add a transition from *decide* to l_0 with guard (increase the speed)
$$d_i \leq d \;\&\&\; d < d_{i-1} \;\&\&\; v < v_i^l$$
- add a transition from l_i to *wait* with update rule:
$$v = \begin{cases} v + a_i \text{ if } v + a_i \geq v_{min} \\ v_{min} \text{ otherwise} \end{cases}$$

The lower bound v_i^l and the upper bound v_i^u partitions $[v_{min}, v_{target}]$ into three regions: $[v_{min}, v_i^l)$, $[v_i^l, v_i^u)$ and $[v_i^u, v_{target}]$. Consider the case when $d \in [d_i, d_{i-1})$. The vehicle accelerates if $v \in [v_{min}, v_i^l)$ (transition to l_0), keeps its speed constant if $v \in [v_i^l, v_i^u)$ (transition to *wait*), and decelerates if $v \in [v_i^u, v_{target}]$ (transition to l_i).

The distance bounds $d_0, d_1, \ldots, d_{m-1}$, and the velocity bounds $v_1^l, v_1^u, \ldots, v_m^l, v_m^l$ are the parameters of the TA

that needs to be set. Whereas, $v_{min}, v_{max}, a_0, \ldots, a_m$ and v_{target} are initialized with respect to the considered system. Note that the guard definitions (transition constraints) guarantee that for any parameter assignment satisfying the ordering defined in (2) and (3), the resulting control TA will be deterministic since the the constraints (guards) on the transitions leaving *decide* are mutually exclusive, and a unique transition leaves every other location. (4) is imposed to form an ordering in the velocity bounds for different distance intervals.

$$d_{i+1} < d_i, \text{ for } i = 0, \ldots, m-1 \tag{2}$$

$$v_i^l < v_i^u \text{ for } i = 1, \ldots, m \tag{3}$$

$$v_{i+1}^l \leq v_i^l, v_{i+1}^u \leq v_i^u \text{ for } i = 1, \ldots, m-1 \tag{4}$$

3.3 The Leading Vehicle Automaton

The TA modeling the leading vehicle is shown in Fig. 4. It has three locations: the initial location *wait* and two committed locations *decide* and *lane_switch*. When it receives the *wake_lead* signal from the distance TA, it randomly accelerates, decelerates (within the speed limits) or maintains its speed and goes back to *wait* through *decide*. If it receives *new_lead* signal, it sets its speed to a value from $[vmin, vmax]$ randomly through *lane_switch* mimicking a vehicle switching to the same lane as the controlled vehicle.

In general, each $u \in \mathbb{U}$ is represented through a transition from *decide* to *wait* with the corresponding speed update, e.g. $v_l = v_l + u$. There is no guard defined over these transitions, thus one of them is picked non-deterministically to model the uncertainty over the lead vehicle's behavior.

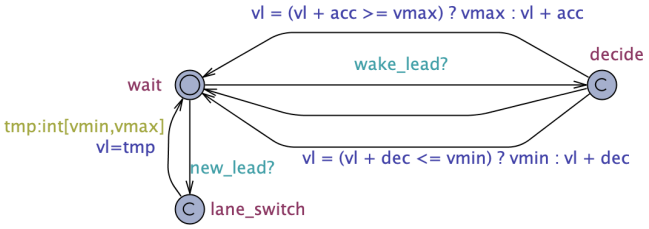

Fig. 4. Lead TA: Models the leading vehicle.

The following parameters of the TA models are initialized with respect to the considered system:

$$d_{range}, d_{lane}, v_{target}, v_{min}, v_{max}, a_0, a_1, \ldots, a_m$$

Whereas, the distance thresholds $d_0, d_1, \ldots, d_{m-1}$ and the velocity thresholds $v_1^l, v_1^u, \ldots, v_m^l, v_m^l$ from the control automaton should be defined. Our goal is to find these parameters such that the resulting system optimizes the driver comfort and satisfies the specifications which are formalized in the next section.

3.4 Formal Specifications

The specifications are expressed in Computation Tree Logic (CTL). The ACC specifications given in Sec. 2.2 are described by the following formula

$$\Phi_{ACC} = \Phi_{safe} \wedge \Phi_{target} \wedge \Phi_{deadlock} \wedge \Phi_{limits} \tag{5}$$

Φ_{safe} encodes the safety property, it requires that $d >= d_{min}$ is always true:

$$\Phi_{safe} = \forall\Box d >= d_{min} \tag{6}$$

Φ_{target} encodes the liveness property, it indicates that it should be possible to reach the target speed:

$$\Phi_{target} = \exists \diamond v = v_{target} \tag{7}$$

Φ_{limits} encodes the restrictions over the speed:

$$\Phi_{limits} = \forall\Box(v_{min} \leq v \wedge v \leq v_{max}) \tag{8}$$

Finally, the last one $\Phi_{deadlock}$ is a commonly used formula to check the modeling errors. It ensures that the system does not stuck at a state, thus a deadlock does not occur:

$$\Phi_{deadlock} = \forall\Box \text{not deadlock} \tag{9}$$

4. PARAMETER SYNTHESIS

The proposed TA models and formal specifications reduce the ACC design problem from Sec. 2 into a parameter synthesis problem for timed automata. While the search space is bounded, the TA does not belong to the L/U class since parameters are used as both upper and lower bounds, e.g, see d_1. In this section, we first define the solution space and enumerate each element in it, which leads to a greedy solution. Then, we define optimization criteria to improve the driver comfort. Finally, we present an efficient iterative heuristic approach to find the optimal parameters.

The search space for the parameter synthesis problem is defined in (10), i.e., the product of the search spaces over the distance thresholds and the speed thresholds satisfying the ordering constraints.

$$\mathbf{S} = \{(\mathbf{d}, \mathbf{v}) = ((d_0, \ldots, d_{m-1}), (v_1^l, v_1^u, \ldots, v_m^l, v_m^l)) \mid \tag{10}$$

$$d_i \in \{d_{min}, d_{min}+1 \ldots, d_{range}\} \text{ for each } i \in \{0, \ldots, m\},$$
$$v_i^l, v_i^u \in \{v_{min}, \ldots, v_{target}\} \text{ for each } i \in \{0, \ldots, m\},$$
$$(\mathbf{d}, \mathbf{v}) \text{ satisfies (2) and (3)}\} \tag{11}$$

Note that the parameters are bounded and takes integer values, thus the search space is finite [3]. In particular, the cardinality of $\mathbf{S}$ is

$$\mid \mathbf{S} \mid = \binom{d_{range} - d_{min}}{m} \binom{v_{target} - v_{min} + m + 1}{2m}, \tag{12}$$

where $\binom{N}{k}$ denotes the number of k-combinations of a set with N elements. As $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$ satisfies the ordering constraint, the corresponding control TA (e.g see Fig. 3) is deterministic. However, it might not satisfy the specification formula Φ_{ACC} (5). The following greedy synthesis approach finds the set of parameter valuations $Sat \subseteq \mathbf{S}$ satisfying the specification. Let $\mathbf{T}$ denote the overall parametric timed automata model (i.e. consists of three automata), and for a given parameter valuation $(\mathbf{d}, \mathbf{v})$, let $\mathbf{T}(\mathbf{d}, \mathbf{v})$ denote the TA initialized with parameters $(\mathbf{d}, \mathbf{v})$.

GreedyApproach:
Initialize $Sat = \emptyset$
For each $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$,
· Model check the $\mathbf{T}(\mathbf{d}, \mathbf{v})$ against Φ_{ACC}
If satisfied, add $(\mathbf{d}, \mathbf{v})$ to Sat

As stated in Sec. 2.2, the objective is to choose the parameter valuation among Sat that optimizes the driver comfort. The driving experience can be improved by (1)

[3] This is a limitation imposed by UPPAAL. It can be mitigated via scaling.

avoiding frequent switches between acceleration levels, (2) maintaining the target speed as much as possible and (3) avoiding sharp decelerations. Due to the non-deterministic nature of the overall TA model because of the lead vehicle, a worst case scenario optimization can be performed. However, the listed criteria conflicts with each other. For example, increasing d_0 reduces the time driven at the target speed (thus conflicts with (2)) and decreases the time spent in a lower deceleration level (improves (3)). A possible approach is to optimize the weighted sum of these criteria. Note that the greedy approach is guaranteed to find the optimal solution for any given optimization function. However, it is computationally very expensive. Next, we present an iterative approach that uses the monotonicity of the parameters to iteratively optimize each parameter in an efficient way. The variations in the parameter optimization step generate strategies optimizing different criteria.

Analysis of Φ_{target} (7): The model construction guarantees that $d_0 \leq d_{range}$ and the transition to the a_0 location (acceleration) is taken whenever $d \geq d_{range}$. Thus, for any $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$, $\mathbf{T}(\mathbf{d}, \mathbf{v})$ satisfies Φ_{target}.

Analysis of Φ_{limits} (8): The update rules for the speed in control TA guarantees that $v \in [v_{min}, v_{target}]$ at all times.

Analysis of $\Phi_{deadlock}$ (9): The property is satisfied when each state of the timed transition system generated from the TA has a successor state (Alur, 2015). Due to the ordering property of the constraint thresholds (10), for any $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$, $\mathbf{T}(\mathbf{d}, \mathbf{v})$ satisfies $\Phi_{deadlock}$.

Thus, the goal is to find a parameter valuation $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$ satisfying Φ_{safe} while optimizing the driver comfort. The proposed approach starts from the most strict parameter valuation $(\mathbf{d}^s, \mathbf{v}^s)$ defined in (13).

$$(\mathbf{d}^s, \mathbf{v}^s) \in \mathbf{S} \text{ where} \tag{13}$$
$$\mathbf{d}^s = (d_{range}, d_{range} - 1, \ldots, d_{range} - m + 1)$$
$$\mathbf{v}^s = (v_1^l, v_1^u, \ldots, v_m^l, v_m^l) \text{ with}$$
$$v_i^l = v_{min}, \text{ and } v_i^u = v_{min} + 1 \text{ for each } i \tag{14}$$

The parameter valuation (13) is strict in the sense that it generates the most aggressively decelerating policy from $\mathbf{S}$. In particular, when $d < d_{range}$, it decelerates with the lowest possible $a \in \mathbb{U}$ allowed by the ordering constraints. Furthermore, due to the $\mathbf{v}^s$ definition, from *decide* location, only the transitions to the locations associated with deceleration (l_i, $i > 0$) can be taken when $d < d_{range}$ and $v \neq v_{min}$. The proposed approach iteratively relaxes the thresholds from $(\mathbf{d}^s, \mathbf{v}^s)$ by using the monotonicity of the parameters, which is defined next.

Monotonicity for the distance parameters: Consider two parameter valuations $(\mathbf{d}^a, \mathbf{v})$ and $(\mathbf{d}^b, \mathbf{v})$ satisfying that $d_i^a < d_i^b$ for some $i \in \{0, \ldots, m\}$ and $d_j^a = d_j^b$ for each $j \neq i$. Then, if $\mathbf{T}(\mathbf{d}^a, \mathbf{v})$ satisfies Φ_{safe}, then $\mathbf{T}(\mathbf{d}^b, \mathbf{v})$ also satisfies Φ_{safe}.

The property indicates if two parameter valuations are the same except the $i-th$ distance parameter, then if the one with a lower distance threshold satisfies the safety specification, then the other one also satisfies it. The property follows from the speed restrictions (4) that $\mathbf{T}(\mathbf{d}^b, \mathbf{v})$ starts decelerating at a higher rate (for a_{i+1}) earlier than $\mathbf{T}(\mathbf{d}^a, \mathbf{v})$. Furthermore, for parameter valuations $(\mathbf{d}^a, \mathbf{v})$ and $(\mathbf{d}^b, \mathbf{v})$, if $\mathbf{T}(\mathbf{d}^b, \mathbf{v})$ does not satisfy Φ_{safe}, then $\mathbf{T}(\mathbf{d}^a, \mathbf{v})$ does not satisfy Φ_{safe} as well, which is utilized to perform a binary search.

Monotonicity for the speed upper bound: Consider two parameter valuations $(\mathbf{d}, \mathbf{v}^a)$ and $(\mathbf{d}, \mathbf{v}^b)$ satisfying that $v_i^{u,a} < v_i^{u,b}$ for some $i \in \{1, \ldots, m\}$ and $v_j^{u,a} = v_j^{u,b}$ for each $j \neq i$, and $v_j^{l,a} = v_j^{l,b}$ for each $j \in \{1, \ldots, m\}$. Then, if $\mathbf{T}(\mathbf{d}, \mathbf{v}^b)$ satisfies Φ_{safe}, then $\mathbf{T}(\mathbf{d}, \mathbf{v}^a)$ also satisfies Φ_{safe}.

The property indicates if two parameter valuations are the same except the $i-th$ speed upper bound, then if the one with a higher bound satisfies the safety specification, then the other one also satisfies it.

Monotonicity for the speed lower bound: Consider two parameter valuations $(\mathbf{d}, \mathbf{v}^a)$ and $(\mathbf{d}, \mathbf{v}^b)$ satisfying that $v_i^{l,a} < v_i^{l,b}$ for some $i \in \{1, \ldots, m\}$ and $v_j^{l,a} = v_j^{l,b}$ for each $j \neq i$, and $v_j^{u,a} = v_j^{u,b}$ for each $j \in \{1, \ldots, m\}$. Then, if $\mathbf{T}(\mathbf{d}, \mathbf{v}^b)$ satisfies Φ_{safe}, then $\mathbf{T}(\mathbf{d}, \mathbf{v}^a)$ also satisfies Φ_{safe}.

Algorithm 1 *ParameterOptimization*

1: $(\mathbf{d}, \mathbf{v}) = (\mathbf{d}^s, \mathbf{v}^s)$ from (13)
2: **return** $\emptyset$ if $\mathbf{T}(\mathbf{d}, \mathbf{v})$ does not Φ_{safe}
3: $d_m = d_{min}$, $v_0^u = v_{target}$, $v_0^l = v_{target}$ (edge conditions)
4: $(\mathbf{d}^p, \mathbf{v}^p) = 0$ (initialization)
5: **while** $(\mathbf{d}, \mathbf{v}) \neq (\mathbf{d}^p, \mathbf{v}^p)$ **do**
6: **while** $i = m$ to 1 **do**
7: $(\mathbf{d}^p, \mathbf{v}^p) = (\mathbf{d}, \mathbf{v})$)
8: $Next$(Dist-i, $\mathbf{T}, (\mathbf{d}, \mathbf{v}), d_{i-1}, d_i, \Phi_{safe}$) ▷ $Next(p, \mathbf{T}, (\mathbf{d}, \mathbf{v}), c, b, \Phi)$ finds a parameter valuation for p within bounds c and b such that $\mathbf{T}(\mathbf{d}, \mathbf{v})$ satisfies Φ.
9: $Next$(**v-u-i**, $\mathbf{T}, (\mathbf{d}, \mathbf{v}), v_i^u, v_{i-1}^u, \Phi_{safe}$)
10: $Next$(**v-l-i**, $\mathbf{T}, (\mathbf{d}, \mathbf{v}), v_i^l, min(v_i^u, v_{i-1}^l), \Phi_{safe}$)
11: **end while**
12: **end while**
13: **return** $(\mathbf{d}, \mathbf{v})$

The proposed optimization method is summarized in Alg. 1. First, it checks whether $\mathbf{T}(\mathbf{d}^s, \mathbf{v}^s)$ satisfies the specification. By the monotonicity properties of the parameters, if $\mathbf{T}(\mathbf{d}^s, \mathbf{v}^s)$ does not satisfy the specification, then no $(\mathbf{d}, \mathbf{v}) \in \mathbf{S}$ satisfies it. If $\mathbf{T}(\mathbf{d}^s, \mathbf{v}^s)$ satisfies Φ_{safe}, then starting from the lowest acceleration level (highest deceleration rate), for each level, *the next* distance threshold, speed upper bound and speed lower bound are found in this order. Note that for each of the optimization steps (lines 8,9,10), a single parameter is found the by virtue of the monotonicity analysis. The main loop continues as long as there is a progress in at least one of the parameters(line 5). The speed bounds creates a buffer zone between the different acceleration levels, thus relaxing speed bounds (line 9,10) avoids frequent switches between the acceleration levels contributing to the first criteria.

Two approaches are used to find the next satisfying valuation. The first one is to use binary search within the given lower and upper bounds (e.g. d_{i-1}, d_i). In this case, the tightest valuation is found for each parameter

in the order in which the optimization is performed, i.e, $d_{m-1}, d_{m-2}, \ldots, d_0, v_1^u, v_1^l, v_2^u, v_2^l, \ldots, v_m^u, v_m^l$. The order for the speed bounds is reversed due to the initial condition and constraint (4). This approach finds the optimal parameter valuation for avoiding higher deceleration rates.

Algorithm 2 $FindNext(\mathbf{T}, (\mathbf{d}, \mathbf{v}), c, b, param, \Phi)$

1: **repeat**
2: $b = (c + b)/2$ (floor for when $b > c$, i.e., maximization, ceiling for minimization)
3: Set $param$ to b in $(\mathbf{d}, \mathbf{v})$
4: **until** $\mathbf{T}(\mathbf{d}, \mathbf{v})$ satisfies Φ

The second approach to find the next satisfying valuation is shown in Alg. 2. Essentially, it is a modified binary search, where the search is terminated when a satisfying value is found. This heuristic approach iteratively relaxes each parameter to find a smooth policy.

4.1 Parameter Synthesis for the Example System

In this section, we present the numerical results for the system introduced in Sec. 2.3. The safety requirement is $\Phi_{safe} = \forall\Box d \geq 15$. The parameters of the model are $(d_0, d_1), (v_1^l, v_1^u, v_2^l, v_2^u)$. The most strict parameter valuation is $((150, 149), ((10, 11, 10, 11))$ (see control TA in Fig. 3). The overall system with these parameters satisfies Φ_{safe}, thus Φ_{ACC} (5) which is validated via UPPAAL.

First, binary search is used to find the tightest valuations in lines 8,9,10. In the first step, for the minimization of d_1, $((150, 15), ((10, 11, 10, 11))$ is found to be satisfying. For this valuation, $d_1 = d_{min}$, thus the control automaton completely avoids $a_2 = -2$. Note that for v_2^l, v_2^u, the upper bounds with respect to (4) are 10 and 11, respectively. Thus their value does not change in lines 9 and 10. Next, the minimization of d_0 returns 70. The optimization of v_1^l, v_1^u for $((70, 15), ((10, 11, 10, 11))$ reveals that increasing any of them results in violation of the specification. Thus the algorithm returns $((70, 15), ((10, 11, 10, 11))$. The resulting strategy is to decelerate at $a_1 = -1$ when there is a vehicle within 70m. When there is a lead vehicle with a constant speed of $v_l = 10m/s(v_{min})$, this strategy matches the speed of the leading vehicle when $d = 15$m.

Next, $FindNext(\cdot)$ method given in Alg. 2 is used in lines 8,9,10. Again, the optimization starts with $((150, 149), ((10, 11, 10, 11))$. The satisfying parameters found in the algorithm are given below. Each line is marked with the iteration number over the main loop and the optimized parameter. The new value is highlighted in blue.

$$
\begin{aligned}
&It: 1, d_1 : ((150, 82), (10, 11, 10, 11))\\
&It: 1, d_0 : ((116, 82), (10, 11, 10, 11))\\
&It: 1, v_1^u : ((116, 82), (10, 15, 10, 11))\\
&It: 1, v_1^l : ((116, 82), (12, 15, 10, 11))\\
&It: 2, d_1 : ((116, 48), (12, 15, 10, 11))\\
&It: 2, d_0 : ((82, 48), (12, 15, 10, 11))\\
&It: 2, v_1^u : ((82, 48), (12, 17, 10, 11))\\
&It: 2, v_1^l : ((82, 48), (15, 17, 10, 11))\\
&It: 3, d_1 : ((82, 32), (15, 17, 10, 11))\\
&It: 3, d_0 : ((57, 32), (15, 17, 10, 11))\\
&It: 4, d_1 : ((57, 31), (15, 17, 10, 11))\\
&It: 4, d_0 : ((54, 31), (15, 17, 10, 11))
\end{aligned}
$$

The resulting ACC strategy uses both of the deceleration levels. Independent of the vehicle's velocity, when $d \leq 31m$, the vehicle decelerates with -2 since $v_2^u = v_{min} + 1$. When $d \geq 54$ the vehicle accelerates to its target speed. Finally, when $d \in [32, 54]$, the vehicle decelerates with -1 if $17 \leq v$, maintains its speed if $15 \leq v < 17$, and accelerates if $v < 15$. This strategy postpones the deceleration compared to the first one (d_0 was 70) by utilizing the second deceleration level.

5. CONCLUSION

We presented a timed automata model for the adaptive cruise control problem. We first constructed a template parametric TA for the given number of acceleration levels. Then, we presented an efficient parameter synthesis method for the particular parametric TA such that the resulting TA was guaranteed to satisfy the specifications.

REFERENCES

(2005). Uppaal. URL `http://www.uppaal.org`.

Alur, R. (2015). *Principles of Cyber-Physical Systems*. The MIT Press.

Beneš, N., Bezděk, P., Larsen, K.G., and Srba, J. (2015). Language emptiness of continuous-time parametric timed automata. In M.M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann (eds.), *Automata, Languages, and Programming*, 69–81. Springer Berlin Heidelberg, Berlin, Heidelberg.

Étienne, A. (2019). What's decidable about parametric timed automata? *Int J Softw Tools Technol Transfer*, 21, 203–219.

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. (2002). Linear parametric model checking of timed automata. *The Journal of Logic and Algebraic Programming*, 52-53, 183 – 220.

ISO (2010). *Intelligent Transport Systems-Adaptive Cruise Control Systems- Performance Requirements and Test Procedures*. ISO Standard 15622:2010 (E).

Jing Zhou and Huei Peng (2005). Range policy of adaptive cruise control vehicles for improved flow stability and string stability. *IEEE Transactions on Intelligent Transportation Systems*, 6(2), 229–237.

Larsen, K.G., Mikučionis, M., and Taankvist, J.H. (2015). *Safe and Optimal Adaptive Cruise Control*, 260–277. Springer International Publishing, Cham.

Magdici, S. and Althoff, M. (2017). Adaptive cruise control with safety guarantees for autonomous vehicles. *IFAC-PapersOnLine*, 50(1), 5774 – 5781. 20th IFAC World Congress.

Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A.D., Grizzle, J.W., Ozay, N., Peng, H., and Tabuada, P. (2016). Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4), 1294–1307.

SAE J2399 (2014). *Adaptive Cruise Control (ACC) Operating Characteristics and User Interface*.

Xiao, W. and Belta, C. (2019). Control barrier functions for systems with high relative degree. In *IEEE Conference on Decision and Control*, 1–6.