A Linear Temporal Logic Based Approach for Vehicle Motion Planning

Dule Shu
Department of Electrical Engineering
The Pennsylvania State University
University park, Pennsylvania 16802
Email: djs6106@psu.edu

Constantino M. Lagoa
Department of Electrical Engineering
The Pennsylvania State University
University park, Pennsylvania 16802
Email: lagoa@psu.edu

***Abstract*—We consider a vehicle path planning problem under the model of a finite transition system. We start with a continuous model for the motion planning problem. By partitioning the state space of the continuous model, we formulate the motion planning problem as a problem of designing switching controller for the finite transition system. We use linear temporal logic (LTL) formulas to describe the behavior of the transition system and the objective of motion planning. Using the LTL planning software toolbox: TuLip, we obtain a switching controller that guarantees the satisfaction of all the LTL formulas we have defined. Under such switching strategy, the vehicle will always avoid colliding with other vehicles while eventually recovering to a desired constant velocity.**

I. Introduction

Advanced driver assistance system (ADAS), a system designed to improve the driving process of vehicles, has been drawing increasing attention in recent years from researchers, automobile manufacturers and vehicle drivers. ADAS technology is expected to enhance vehicle safety and reduce drivers' workload. For vehicle safety, ADAS either warns the driver of imminent collisions [1], or directly avoids them by interfering the control of the vehicle [2], [3]. To reduce the workload of drivers, ADAS uses adaptive cruise control [4], autonomous driving [5] and other techniques to automatically adjust the states of the vehicle in response to the changing environment of road traffic.

In terms of control systems, implementing ADAS usually requires a group of controllers organized in a hierarchical structure. For example, at the higher level of the hierarchy, there is a supervisory controller that determines in which mode the vehicle is operated [6]. The mode can be automatic navigation with longitudinal control [7], emergency lateral maneuvers [8], or manual control by the driver; at the intermediate level, there is a motion planning controller that generates a desired trajectory or a desired speed for the vehicle to follow [9], [10]; and at the lower level, there are controllers that control the thrust and torque on the vehicle in order to reach a desired velocity or achieve lane keeping [11], [12]. The hierarchical structure of controllers provides a well-ordered design scheme for the overall control system, and makes the design of each individual controller relatively simple. On the other hand, hierarchical structure may contain the following three problems: 1. there may not be a mathematical evaluation for the performance of the supervisory controller. If this is true, then even if lower level controllers are guaranteed to be stable through controller design, the performance of the overall hierarchical structure is still mathematically undetermined. 2. The switching between two modes may cause instability of the overall system, if designers only focus on satisfying the individual performance of each controller. 3. The complexity of the hierarchical structure may result in high demand for the computation power of the vehicle electronic system and therefore increases the cost of ADAS technology.

Motivated by the popularity of ADAS and the three potential problems of the hierarchical control system mentioned before, we propose a method for vehicle motion planning using linear temporal logic (LTL). By developing this approach, we would like to provide a theoretical framework that addresses the first and the third problems. In addition, our method can interface with existing algorithms in [13] and [14] to fix the second problem. Our motion planning method is a switching control strategy that guides the vehicle in a partitioned space of the road. The motion of the vehicle in the partitioned space is modeled as a finite state transition system. LTL is used to describe the behavior of the transition system and the objective of motion planning. Using a Python-based planning software toolbox named TuLip [15], we can compute a switching strategy that is mathematically guaranteed to achieve the motion planning objective. The contribution of the paper is as follows: We introduce an LTL-based approach for vehicle motion planning. Different from the LTL-based approaches in [14], [16], [17], our approach defines the environment variables as the states of the target vehicles and uses a local coordinate frame. Our tentative approach provides a novel perspective in analyzing the vehicle motion planning problem. From this perspective, relationship between the host vehicle and the target vehicles is considered as a two player game described by LTL formulas. The potential advantage of using LTL-based approach is that the motion planning algorithm computed by this approach requires less workload of the onboard ADAS. The reduction in workload is achieved in two ways. First, our approach exhaustively computes all possible driving situations offline based on an abstract traffic model. Though the computation is time-consuming, it only needs to be done once. Then, the computation result can be uploaded to

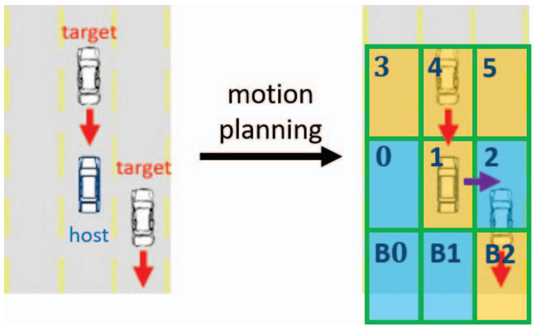

Fig. 1. Example problem of motion planning (driving direction: up)

the ADAS of the vehicle as a look-up table, which makes the real time motion planning computationally efficient. Second, since our approach plans vehicle motion in a local coordinate frame fixed to the host vehicle, GPS is not needed to measure the global position of the host vehicle.

Essentially, our proposed approach is designed to deal with traffic situations such as the one shown in the left part of Figure 1. For a vehicle (named host vehicle) driving on the road, if both a vehicle in the front and a vehicle on its right (named target vehicles) are decelerating, what should an automatic pilot of the host vehicle do in response? To answer this question, our approach checks an area around the host vehicle, partitions the area into multiple cells (as marked by green boxes), determines which cells can be entered (as colored with blue) and which cannot (as colored with yellow). As will be shown in Section IV, our approach concludes that the host vehicle should switch to the right lane while maintaining its longitudinal velocity (as shown in the right part of Figure 1). Although the example scenario seems too trivial to be a research problem, what we are really interested in is to establish a theoretical framework which is capable of solving a more general and complicated problem with guaranteed performance if the solution exists. The problem is solved offline using the language of math logic, so that the real-time execution of the approach requires much less computational resource.

II. Preliminaries

A. Finite-state approximation

Consider nonlinear systems of the following form

$$x(k+1) = f(x(k), d(k)) \tag{1}$$

where k is the time step, $x(k) \in V \subset \mathbf{R}^n$ is the state at time k, $d(k) \in W \subset \mathbf{R}^l$ is the exogenous disturbance. The control input $u(k)$ is implicitly included in (1). For example, $u(k)$ can appear as a closed loop controller $u(k) = K(x(k))$.

In a system represented by (1), both the values of $x(k)$ and $d(k)$ are continuous. To apply linear temporal logic in controller design, we need to convert (1) to a discrete valued system. The discrete valued system is a finite state approximation of the original continuous valued system. It is obtained by partitioning the spaces $\mathbf{R}^n$ and $\mathbf{R}^l$ in (1).

To properly define the partition, we introduce a map from the continuous space to a set of discrete values. We define $T_x : \mathbf{R}^n \to \mathcal{P}$ which maps each value of the state $x(k)$ to an element of a finite set $\mathcal{Q} := \{q_0, q_1, \dots, q_r\}$. Similarly, we define $T_d : \mathbf{R}^l \to \mathcal{E}$ which maps each value of the disturbance $d(k)$ to an element of a finite set $\mathcal{E} := \{e_0, e_1, \dots, e_p\}$. With the definition of partition, we can properly define the finite transition system as follows.

Definition 1. A finite transition system is a tuple: $\mathcal{G} := \{\mathcal{Q}_0, \mathcal{Q}, \to\}$, where $\mathcal{Q}_0 \subset \mathcal{Q}$ is a set of initial values of states, and $\to := \{(q_i, q_j) \mid q_i, q_j \in \mathcal{Q}\}$ denotes the transition from q_i to q_j.

Compared with the more comprehensive definition of transition systems in [18], Definition 1 omits the concepts of actions, atomic propositions and labeling functions. This is because at this moment, we have not introduced the concept of math logic yet, and we only want to show the basic structure of a transition system. Those omitted concepts will appear in the later part of the paper.

Note that for arbitrary q_i and q_j, it is possible that q_j cannot be reached from q_i. Further discussion on such reachability problem can be found in [14]. Since the focus of this paper is the design of switching controller for the finite transition system, we assume that the transition $q_i \to q_j$ always exists if it is defined in our transition system model.

B. Linear Temporal Logic

Given a finite state transition system, we can describe some of its properties using the language of linear temporal logic (LTL). Those properties described by LTL are called specifications. LTL is an extension of propositional logic. Propositional logic is a branch of math logic which is built on propositions, e.g., $\alpha, \beta, \gamma, \dots$ and logic connectives, e.g., negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$)... For example, if two propositions α and β are defined as follows. $\alpha :=$ ”*The vehicle is driving at the desired speed*”, $\beta :=$ ”*The vehicle is driving in the right lane*”. Then a third proposition φ can be defined as $\varphi := \alpha \wedge \neg\beta$, which means the vehicle is driving at the desired speed and is not in the right lane. LTL extends propositional logic by introducing extra temporal logic operators including next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$).

Formally, LTL formulas are defined inductively over a set of atomic propositions (propositions that do not contain other propositions) using the following formula:

$$\varphi := \mathit{True} \mid \alpha \mid \neg\varphi \mid \varphi \vee \psi \mid \bigcirc\varphi \mid \varphi \;\mathcal{U}\; \psi$$

which means an arbitrary LTL formula φ can be constructed with constants (*True*), atomic propositions (α) and the four listed logic connectives ($\neg$, $\vee$, $\bigcirc$ and $\mathcal{U}$). The listed connectives are sufficient for constructing arbitrary formulas, since other logic connectives such as conjuction, implication, eventually and always can be defined using the listed connectives. For example, $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \to \psi := \neg\varphi \wedge \psi$, $\Diamond\varphi := \mathit{True}\;\mathcal{U}\;\varphi$, and $\Box\varphi := \neg\Diamond\neg\varphi$.

Consider a state sequence of a continuous system $x(0)x(1)x(2)x(3)...$ In the corresponding finite state transition system, such sequence is represented by a sequence of discrete states: $p_0p_1p_2...$ We define an LTL specification φ over the sequence $p_0p_1p_2...$ as follows. Let $L : \mathcal{Q} \rightarrow 2^{\mathcal{AP}}$ be a mapping (the labeling function) where $\mathcal{AP}$ is a set of atomic propositions and $2^{\mathcal{AP}}$ is the set of all possible truth values of $\mathcal{AP}$. Then, the LTL specification over state sequence $p_0p_1p_2...$ can be defined inductively as follows.

Definition 2. LTL specification for transition systems

- For a discrete state p_i and atomic proposition α, we write $p_i \models \alpha$ if and only if α is true at position i.
- For p_i and LTL formulas φ and ψ, we write $p_i \models \neg\varphi$ if and only if $p_i \not\models \varphi$, $p_i \models \bigcirc\varphi$ if and only if $p_{i+1} \models \varphi$, $p_i \models \varphi \vee \psi$ if and only if $p_i \models \varphi$ or $p_i \models \psi$, $p_i \models$ *True* if and only if $p_i \models \varphi \vee \neg\varphi$ for any $\varphi \in \mathcal{AP}$, and $p_i \models \varphi\mathcal{U}\psi$ if and only if there exists $j \geq i$ such that $p_k \models \varphi$ for $k \in [i, j)$ and $p_j \models \psi$.
- We write $p_0p_1p_2... \models \varphi$ if and only if $p_0 \models \varphi$.

Our vehicle motion planning problem is essentially a problem of designing switching controller for a finite transition system. That is, given a finite transition system $\mathcal{G}$, a set of discrete environment states $\mathcal{E}$ (in our case, a set of exogenous disturbance), and two LTL formulas φ and ψ, design a switching controller such that the following statement holds.

Let Γ be the set of environment state sequences, and let Σ be the set of state sequences of $\mathcal{G}$. $\Sigma \models \varphi$ if $\Gamma \models \psi$.

In other words, our goal is to develop a switching control strategy under which the transition system always satisfies φ regardless of how environment states change, as long as the change is constrained by ψ.

C. The LTL planning software toolbox: TuLip

In this paper, we use TuLip to compute a path for the vehicle. TuLip is a Python-based toolbox for embedded control software synthesis [15]. In general, TuLip can be used to design a controller for a continuous-state linear time-invariant system. To do this, TuLip first creates a finite state transition system by discretizing the continuous state space. Then, it computes a switching strategy that satisfies users specification in LTL formulas. Finally, a continuous controller is synthesized for the original continuous model.

III. Problem Formulation

A. Transition system model for vehicle motion planning

To define the continuous-valued state space of the vehicle, we start with the point mass model used in [19].

$$\begin{aligned}\dot{x} &= v\cos\theta\\ \dot{y} &= v\sin\theta\\ \dot{\theta} &= \frac{v}{M}u_1\\ \dot{v} &= u_2\end{aligned} \tag{2}$$

All variables in (2) are defined with respect to the road-fixed frame. x and y are the vehicle positions in forward and lateral directions, respectively. M is the mass of the vehicle. θ is the heading angle. v is the linear velocity of the vehicle. u_1 and u_2 are the steering control input and the acceleration control input, respectively. We assign the subscript "*s*" to any variable in (2) to indicate that such variable is associated with the host vehicle. Similarly, the subscript "*ei*" is assigned to any variable associated with the i-th target vehicle.

Then, we consider a different coordinate frame named $x'o'y'$ for the vehicle motion planning problem. In $x'o'y'$, we use $x'_s$and y'_s to represent the position of the host vehicle. We let $x'_s \equiv 0$, which indicates that $x'o'y'$ is always fixed to the body of the host vehicle in the direction of x' axis. In addition, we let $y'_s \equiv y_s$, which means that $x'o'y'$ is fixed to the road in the direction of y' axis. Thus, in the direction of x' axis, the position of the i-th target vehicle remains as y_{ei}; and in the direction of of y' axis, the position of the i-th target vehicle is changed to $x_{ei} - x_s$. In addition, we add the following definitions to our problem formulation:

$$p_s := y_s,\ v_s := \begin{pmatrix} \dot{x}_s & \dot{y}_s \end{pmatrix}^T,$$

$$p_{ei} := \begin{pmatrix} x_{ei} - x_s & y_{ei} \end{pmatrix}^T,\ d_i := \begin{pmatrix} \dot{x}_{ei} - \dot{x}_s & \dot{y}_{ei} \end{pmatrix}^T.$$

Then, our continuous state space is defined as follows.

$$x := \begin{pmatrix} p_s & v_s & p_{e1} & p_{e2} & \dots & p_{ek} \end{pmatrix}^T \in \mathbf{V} \subseteq \mathbf{R}^{3+2k},$$

$$d := \begin{pmatrix} d_1 & d_2 & \dots & d_k \end{pmatrix}^T \in \mathbf{W} \subseteq \mathbf{R}^{2k}$$

where the vector space $\mathbf{V}$ consists of the position and velocity of the host vehicle and the positions of the k target vehicles, and the vector space $\mathbf{W}$ consists of the velocities of the k-th target vehicle.

To partition the continuous state spaces, we consider the following maps: $T_x : \mathbf{R}^{3+2k} \rightarrow \mathcal{Q}$ and $T_d : \mathbf{R}^{2k} \rightarrow \mathcal{E}$ which map the continuous spaces to some finite sets $\mathcal{Q}$ and $\mathcal{E}$. We define $\mathcal{Q}$ as $\mathcal{Q} = \mathcal{S} \times \mathcal{T}_1 \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_k$ where $\mathcal{S}$ is the finite set of host vehicle positions and velocities, and $\mathcal{T}_i (i = 1, \ldots, k)$ is the finite set of positions of the i-th target vehicle. We define $\mathcal{E} = \mathcal{E}_1 \times \mathcal{E}_2 \times \ldots \times \mathcal{E}_k$ where $\mathcal{E}_i (i = 1, \ldots, k)$ is the finite set of target vehicle velocities. To specify $\mathcal{S}, \mathcal{T}_i$ and $\mathcal{E}_i$, we consider the following area of interest in the $x'o'y'$ frame.

As shown in Figure 2, we partition the area into $m \times n$ cells, then assign a natural number to each of the cells. Thus, the set $\{0, 1, \ldots, mn - 1\}$ becomes the finite state approximation of the vehicle position. For the vehicle velocity, we consider the following set as the finite state approximation: $\{$*constant*, *left*, *right*, *backward*, *forward*$\}$. "*Constant*" means that the vehicle is driving along the x' axis at a fixed desired velocity set by the host vehicle driver. "*Left*" means that the vehicle is driving at the desired velocity along the x' axis, and meanwhile has a fixed positive velocity along the y' axis. Similar to "*Left*", "*Right*" means that the vehicle is driving at the desired velocity along the x' axis, and meanwhile has a fixed negative velocity along the y' axis. "*Backward*" means that the vehicle is driving along the x' axis at a fixed velocity that is lower than the desired velocity. Finally, "*Forward*" means that the vehicle is

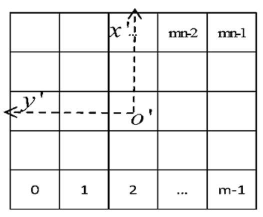

Fig. 2. Partitioning of road space of the vehicles

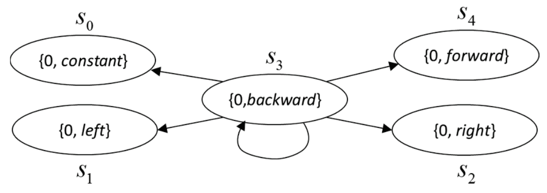

Fig. 3. Possible transitions from s_3

driving along the x' axis at a fixed velocity that is higher than the desired velocity.

With the finite state approximation of position and velocity, we define the discrete state of our transition system and the discrete state of exogenous disturbance as follows.

$$\begin{aligned} &\mathcal{Q} = \mathcal{S} \times \mathcal{T}_1 \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_k \\ &\mathcal{S} = \{0, 1, \ldots, mn-1\} \times \\ &\{\textit{constant}, \textit{left}, \textit{right}, \textit{backward}, \textit{forward}\} \\ &\mathcal{T}_i = \{0, 1, \ldots, mn-1\} \\ &\mathcal{E}_i = \{\textit{constant}, \textit{left}, \textit{right}, \textit{backward}, \textit{forward}\} \\ &i \in \{1, 2, \ldots, k\} \end{aligned} \tag{3}$$

As defined in (3), $\mathcal{S}$ consists of the pairs of position and velocity of the host vehicle, $\mathcal{T}_i$ consists of the positions of the i-th target vehicle, and $\mathcal{E}_i$ consists of the velocities of the i-th target vehicle.

In our transition system model $\mathcal{G} := \{\mathcal{Q}_0, \mathcal{Q}, \rightarrow\}$ for the vehicle motion planning problem, we have defined $\mathcal{Q}$ in (3), and we still need to define the transition relation $\rightarrow$. We mainly use equations (6) to (9) to specify a state transition $q(k) \rightarrow q(k+1)$ where $q(k), q(k+1) \in \mathcal{Q}$.

Let $p_x(k) \in \{0, 1, \ldots, m-1\}$ be the position coordinate on the y' axis at step k. Let $p_y(k) \in \{0, 1, \ldots, n-1\}$ be the position coordinate on the x' axis at step k. Let $\alpha, \beta \in \{\textit{constant}, \textit{left}, \textit{right}, \textit{backward}, \textit{forward}\}$ be the velocities of the host and the i-th target vehicle at step k, respectively. Let $h_x, h_y : \{\textit{constant}, \textit{left}, \textit{right}, \textit{backward}, \textit{forward}\} \rightarrow \{0, -1, 1\}$ be two maps defined as follows.

$$\begin{aligned} &h_x(\textit{constant}) = 0,\ h_x(\textit{left}) = 0,\ h_x(\textit{right}) = 0, \\ &h_x(\textit{backward}) = -1,\ h_x(\textit{forward}) = 1, \\ &h_y(\textit{constant}) = 0,\ h_y(\textit{left}) = -1,\ h_y(\textit{right}) = 1, \\ &h_y(\textit{backward}) = 0,\ h_y(\textit{forward}) = 0 \end{aligned} \tag{4}$$

Then, for the target vehicle, the transition of the position state is defined as follows.

$$\begin{aligned} p_y(k+1) &= p_y(k) + h_y(\beta), \\ p_x(k+1) &= p_x(k) + h_x(\beta) - h_x(\alpha) \end{aligned} \tag{5}$$

For the host vehicles, the transition of the position state is defined as follows.

$$p_y(k+1) = p_y(k) + h_y(\alpha), \quad p_x(k) \equiv 0 \tag{6}$$

The position $p(k) \in \{0, 1, \ldots, mn-1\}$ of any vehicle can be calculated using

$$p(k) = m \cdot p_x(k) + p_y(k). \tag{7}$$

When it comes to the case where some vehicles are driving near the boundary of the driving space, more regulations in addition to (6) $-$ (9) need to be introduced. These regulations includes:

1) A target vehicle can enter or leave the driving space through its boundary.
2) The host vehicle cannot leave the driving space.
3) A host vehicle does not appear in a boundary cell if the host vehicle is in that cell.

Remark: The discrete state transition is defined as an approximation of the continuous state transition. In continuous state space, the state transition often yields more general state trajectories in the form of complicated curves. Since the focus of this paper is to show a method of motion planning for the discrete transition system, we have abstracted out those more general trajectories. To model those trajectories, we can introduce more discrete values of velocity and a finer partition of the driving space, although these efforts will increase the computational complexity for motion planning.

B. LTL specifications

After developing the transition system model, we show how to use LTL formulas to describe state transition and the motion planning objective. By converting these LTL formulas into Python code, we can use TuLip to compute a switching strategy which achieves the motion planning objective.

We assign an atomic proposition to each member of $\mathcal{S}, \mathcal{T}_i$, and $\mathcal{E}_i, i \in \{1, 2, \ldots, k\}$. Specifically, we assign s_0 to the host vehicle state $(0, \textit{constant}), s_1$ to the host vehicle state $(0, \textit{right})$, $\ldots$, and s_{5m-3} to the host vehicle state $(m-1, \textit{forward})$. We also assign atomic proposition t_{ij} to the j-th element of $\mathcal{T}_i$. Finally, we assign atomic proposition e_{ij} to the jth element of $\mathcal{E}_i$. The truth values of those atomic propositions indicate the value of the discrete state. For example, if s_0 is true, then the host vehicle is in cell 0 and is driving at the desired constant speed; if t_{ij} is true, then the i-th target vehicle is in the j-th cell; and if e_{i0} is false, then the velocity of the i-th target vehicle is not "*constant*".

Based on the atomic propositions we have defined, we assign a set of LTL formulas to describe the state transitions partly specified by (6) $-$ (9). For example, if the current state of the host vehicle is $s_3 = \{0, \textit{backward}\}$, then its next position will still be 0, and its next velocity is arbitrary. There are five possible transitions from state s_3, as shown in Figure 3. The LTL formula to describe the possible transitions in Figure 3 is

$$s_3 \rightarrow \bigcirc(s_0 \vee s_1 \vee s_2 \vee s_3 \vee s_4). \tag{8}$$

(8) says that if s_3 is true, then in the next step, either s_0, s_1, s_2, s_3 or s_4 is true. If at the same time, the i-th target vehicle is in cell 1 and has a velocity of "*forward*", and the j-th target vehicle is in cell m and has a velocity of "*left*", then we use the following LTL formulas to describe the transition of the positions of the two target vehicles.

$$s_3 \wedge t_{i1} \wedge e_{i0} \rightarrow \bigcirc t_{i,1+2m} \tag{9}$$

$$s_3 \wedge t_{jm} \wedge e_{i1} \rightarrow \bigcirc \neg \bigvee_{r=0}^{mn-1} t_{jr} \tag{10}$$

(9) says that if s_3, t_{i1} and e_{i0} are all true, then in the next step, $t_{i,1+2m}$ is true. $t_{i,1+2m}$ being true indicates that the next position of the i-th target vehicle is two cells ahead of cell m. Similarly, (10) says that if s_3, t_{i1} and e_{i0} all hold, then in the next step, none of t_{j0}, t_{j1}... $t_{j,mn-1}$ hold, which indicates that the j-th target vehicle has left the road.

Next, we specify our motion planning objective in LTL formulas. The objective consists of two parts. The first part is collision avoidance, which requires that the host vehicle does not collide with any target vehicle. The second part is velocity maintenance, which requires that the velocity of the host vehicle becomes forward infinitely often. In other words, for an arbitrary sequence of discrete velocities, e.g., $v_0v_1v_2$, there exists a finite $n \geq 0$ such that $v_n =$"*forward*".

To describe the collision avoidance, we introduce the following formula.

$$\square \bigwedge_{p=1}^{k} \bigwedge_{i=0}^{m-1} \left(\left(\bigvee_{j \in \mathcal{J}_i} s_j \right) \rightarrow \neg t_{pi} \right) \tag{11}$$

where $\mathcal{J}_i$ is the set of indices such that for any $j \in \mathcal{J}_i$, if s_j is true, then the host vehicle is in cell i. (11) requires that any target vehicle cannot be in the same cell where the host vehicle is. To describe the velocity maintenance, we define the following formula

$$\square \lozenge \left(\bigvee_{i \in \mathcal{I}} s_i \right) \tag{12}$$

where $\mathcal{I}$ is the set of indices such that for any $i \in \mathcal{I}$, if s_i is true, then the velocity of the host vehicle is "*forward*". (12) requires that the speed of the host vehicle has to be constant infinitely often.

C. State transition as a two-player game

As mentioned in Section 2, the problem of designing a switching controller is a process of finding a winning strategy for the system in a two-player game. At each transition step of the game, the environment always makes a move first, then the system follows by making its own move and waiting for the next move of the environment. We choose the velocities of the target vehicles as the environment variables, and the system variables comprise of the host vehicle position, the host vehicle velocity, and the target vehicle positions. The state space of the game is $\mathcal{E} \times \mathcal{Q}$. Let $e_i \in \mathcal{E}$ and $q_i \in \mathcal{Q}$ be the environment state and the system state at the i-th step of the game. An execution of the game is

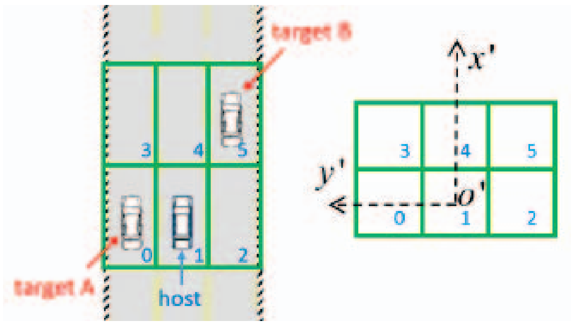

Fig. 4. 3-by-2 partition of the driving space

$$(e_0, q_0) \rightarrow (e_1, q_1) \rightarrow (e_2, q_2) \rightarrow \ldots \rightarrow (e_i, q_i) \rightarrow \ldots$$

The switching strategy of the host vehicle is represented by the sequence of q_i which represents the host vehicle velocity, e.g.,

$$\textit{constant} \rightarrow \textit{backward} \rightarrow \textit{left} \rightarrow \ldots \rightarrow \textit{constant} \rightarrow \ldots.$$

IV. Simulation Results

We implemented the LTL-based motion planning method proposed in Section 3 in TuLip. The driving space is partitioned into six cells, as shown in Figure 4. The driving space contains one host vehicle and at most two target vehicles. The initial velocity of all vehicles are "*constant*". The host vehicle velocity is partitioned into only four discrete values, with "*forward*" velocity excluded for simplicity.

We write the LTL specifications in Python programs and use TuLip to compute a path for the host vehicle. If the specifications are feasible, TuLip will generate a switching control strategy in the form of a finite automaton. Users can define a sequence of target vehicle velocities, and obtain a sequence of system variables that tells how the position and velocity of the host vehicle and the positions of the target vehicle evolve with respect to the target vehicle velocities. We show three motion planning scenarios in Figure 5, where arrows represent a non-"*constant*" velocity, and grey dot represents a target vehicle outside the driving area. Each scenario is illustrated by one subfigure. In the scenario shown by the right subfigure, the two target vehicles keep decelerating for two consecutive steps: $k+1$ and $k+2$. In step $k+1$, the host vehicle chooses to decelerate, while in step $k+2$, the host vehicle chooses to switch to the right lane, as keeping decelerating may violate the motion planning goal of eventually recovering to "*constant*" velocity. Figure 6 shows the truth value of the atomic specification that represents the "*constant*" velocity of the host vehicle, where the upper, middle and lower subplots correspond to the left, middle and right subfigures in Figure 5, respectively. In all three scenarios, the velocity of the host vehicle eventually returns to "*constant*".

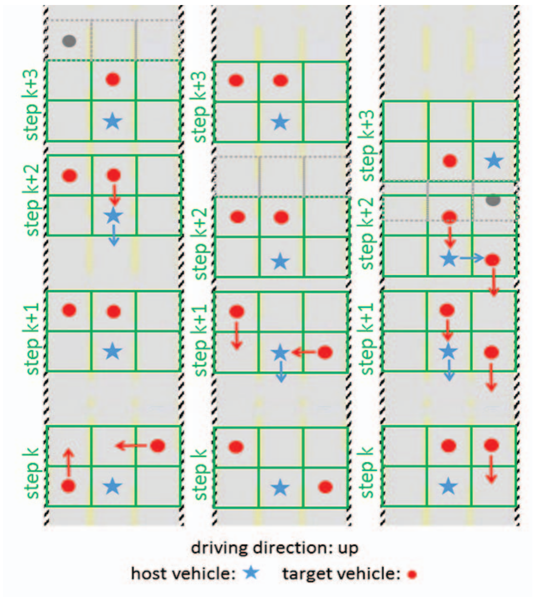

Fig. 5. Simulation result of motion planning in a 3-by-2 cells driving space

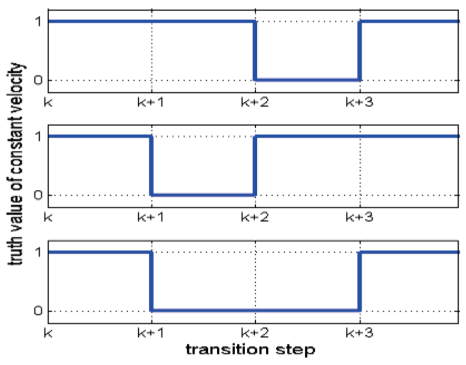

Fig. 6. Truth values of the "*constant*" velocity formula for the three scenarios shown in Figure 5

A. Conclusion

In this paper, we consider a problem of vehicle motion planning which involves a host vehicle and several target vehicles around it. By partitioning an area around the host vehicle, we develop a finite state transition system to model the motion planning problem. We show that the problem can be formulated with LTL formulas. a switching controller is synthesized using the LTL planning software toolbox: TuLip. With this controller, the host vehicle can avoid colliding with the target vehicles while eventually returning to a desired velocity set by the driver. Our future work is aimed on finding a more realistic transition system model while keeping computational complexity within a reseanable amount.

Acknowledgment

We express our sincere thanks to Argonne National Lab and Pennsylvania State University for their sponsorship to our research. We also thank Ms. Jingyao Wang for her help in type setting this paper.

References

[1] H.-S. Tan and J. Huang, "Dgps-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 415–428, 2006.
[2] C.-F. Lin, J.-C. Juang, and K.-R. Li, "Active collision avoidance system for steering control of autonomous vehicles," *IET Intelligent Transport Systems*, vol. 8, no. 6, pp. 550–557, 2014.
[3] F. Jimnez, J. E. Naranjo, and . Gmez, "Autonomous collision avoidance system based on accurate knowledge of the vehicle surroundings," *IET Intelligent Transport Systems*, vol. 9, no. 1, pp. 105–117, 2015.
[4] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, 2003.
[5] G. Stanek, D. Langer, B. Mller-Bessler, and B. Huhnke, "Junior 3: A test platform for advanced driver assistance systems," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. San Diego, CA: IEEE, June 21-24 2010, pp. 143–149.
[6] N. M. Enache, S. Mammar, M. Netto, and B. Lusetti, "Driver steering assistance for lane-departure avoidance based on hybrid automata and composite lyapunov function," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 28–39, 2010.
[7] S. Mammar, N. A. Oufroukh, Z. Yacine, D. Ichalal, and L. Nouvelire, "Invariant set based variable headway time vehicle longitudinal control assistance," in *2012 American Control Conference (ACC)*. Montreal, QC: IEEE, June 27-29 2012, pp. 2922–2927.
[8] R. E. Benton and D. Smith, "A static-output-feedback design procedure for robust emergency lateral control of a highway vehicle," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 618–623, 2005.
[9] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.
[10] J. Hardy and M. Campbell, "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 913–929, 2013.
[11] M. Salazar, A. Alessandretti, A. P. Aguiar, and C. N. Jones, "An energy efficient trajectory tracking controller for car-like vehicles using model predictive control," in *2015 54th IEEE Conference on Decision and Control (CDC)*. Osaka, Japan: IEEE, Dec 15-18 2015, pp. 3675–3680.
[12] M. Canale, L. Fagiano, A. Ferrara, and C. Vecchio, "Comparing internal model control and sliding-mode approaches for vehicle yaw control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 31–41, 2009.
[13] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. Karlsruhe, Germany: IEEE, May 6-10 2013, pp. 5033 – 5040.
[14] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.
[15] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: A software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '11. New York, NY, USA: ACM, 2011, pp. 313–314. [Online]. Available: http://doi.acm.org/10.1145/1967701.1967747
[16] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Switching protocol synthesis for temporal logic specifications," in *2012 American Control Conference (ACC)*. Fairmont Queen Elizabeth, Montreal, Canada: IEEE, Jun 27-29 2012, pp. 727–734.
[17] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Preliminary results on correct-by-construction control software synthesis for adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. Los Angeles, CA: IEEE, Dec 15-17 2014, pp. 816–823.
[18] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.
[19] D. Sadigh and A. Kapoor, "Safe control under uncertainty," Tech. Rep., Oct 2015. [Online]. Available: http://arxiv.org/pdf/1510.07313v1.pdf