Submitted, 2015 Conference on Decision and Control (CDC)
http://www.cds.caltech.edu/~murray/papers/hmm15-cdc.html

Synthesis of Distributed Longitudinal Control Protocols for a Platoon of Autonomous Vehicles

Duo Han, Yilin Mo, Richard M. Murray

***Abstract*—We develop a framework for control protocol synthesis for a platoon of autonomous vehicles subject to temporal logic specifications. We describe the desired behavior of the platoon in a set of linear temporal logic formulas, such as collision avoidance, close spacing or comfortability. The problem of decomposing a global specification for the platoon into distributed specification for each pair of adjacent vehicles is hard to solve. We use the invariant specifications to tackle this problem and the decomposition is proved to be scalable.. Based on the specifications in Assumption/Guarantee form, we can construct a two-player game (between the vehicle and its closest leader) locally to automatically synthesize a controller protocol for each vehicle. Simulation example for a distributed vehicles control problem is also shown.**

I. Introduction

Traffic safety and efficiency have been of primary concern for decades. The automated highway systems [1] which are designed to ensure safety and increase traffic flow at the same time have attracted increasing attention. The main idea is to form a platoon of closely spaced vehicles moving at a high speed. The benefits of a close platoon include greater capacity on the road, weaker impact of vehicles collision (due to close spacing) and less aerodynamic drag [2]. The vehicles (except the leader usually driven by human) have automated control of speed and steering since human drivers cannot react timely to drive safe. The controller of each vehicle mainly relies on the information collected from two sources: one is Vehicle to Vehicle (V2V) communication which share data among the vehicular communication network like DSRC [3]; the other one is the data measured by sensors such as radars or lidars. The difference is that the V2V communication renders a vehicle comprehensive data of other vehicles such as position, velocity and acceleration while the sensors only measure the longitudinal and lateral position of its neighbors.

In this paper we focus on the longitudinal control rather than lateral control. More specifically, we care about maintaining inter-vehicle gap size, collision avoidance, keeping stable speed, etc. A lot of work have been done on longitudinal control protocols for a platoon of vehicles [4]–[10]. Compared to these traditional design-and-verify approaches, we want to specify the desired system behavior and automatically synthesize a controller that is provably correct subject to the system requirements. Temporal logics such as linear temporal logic (LTL) provides such a natural framework to express desired behavior. The advantage of temporal logics over other formalisms like regular language is the resemblance to the natural human language which we will see later. LTL is a powerful task-specification language branch for robotics and automated vehicles to encode the tasks such as safety (A always happens) or liveness (B always eventually happens). The success of LTL specifications have been witnessed in robots motion planning and autonomous vehicles path planning [11]–[14]. The common approach for motion planning with LTL specifications is to construct a finite transition system as the abstraction of the original physical dynamical system [12], [15]. The abstraction describes the possible behavior in finite state space. Based on the abstracted finite transition system and the LTL specifications, a control protocol can be automatically synthesized based on the automaton-based approach.

For a vehicle in the platoon, the behavior of the preceding vehicle and its follower is totally unpredictable when the communication links among them temporarily fail. The only information is the relative longitudinal position. Even if there is communication which tells it the real-time dynamics of its neighbors, the vehicle still has to cooperate with others to satisfy the specification. For example, the vehicle should conservatively slow down a little when links fail, to prevent the sudden brake of its preceding vehicle though it may not occur, to satisfy the safety specification. Thus the vehicle system can be treated as a reactive system since the controller does not only consider the state of its own system which maintains an interaction with the environment but also the state of the environment (i.e., its neighbors and the communication channel status) to satisfy a specification. Our goal is to synthesize a distributed control protocol for each reactive system such that the whole system of the platoon satisfy some given specifications.

The main contributions of this paper are summarized as follows. Firstly, we formulate the distributed control of a platoon of vehicles problem in the linear temporal logic framework. The formalization allows automatic synthesis of the reactive control protocols based on the model-checking theory [16]. Secondly, the distributed synthesis in general undecidable [17]. We attack the difficulty of decomposing a global specification into an equivalent set of distributed specifications by introducing an invariant specification. We can prove the satisfiability of the revised distributed specifications is a sufficient condition for the satisfiability of the global guarantee specifications. By introducing an invariant specification, the synthesis problem is scalable with any number of vehicles in the platoon. Lastly, we provide a specify-and-synthesize design

D. Han is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: dhanaa@ust.hk.

Y. Mo and R. M. Murray are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA. Email: yilinmo@caltech.edu, murray@cds.caltech.edu.

flow demonstration on an example.

The remainder of the paper is organized as follows. Section II presents a technical description of LTL and other formal definitions. Section III formulates the distributed control problem and Section IV reformulates the original problem and gives an alternative approach to synthesize the distributed control protocols. Section V demonstrate the specify-and-synthesize design flow using an example. We conclude with future directions in Section VI.

II. Preliminaries

We present some preliminary materials on finite transition system and linear temporal logic language. The LTL languages are useful in describing more complicated control tasks and we try to formulate the platoon control problem into a two-player game which contains a leader and a follower.

A. Finite transition system

Definition 2.1: An $atomic\ proposition$ is an assertion on a system variable that must be true or false given the value of the variable. If the atomic proposition p is $True$ at the state s, we denote this as $s \vDash p$. Otherwise, we denote it as $s \nvDash p$.

Definition 2.2: A $finite\ transition\ system$ is a tuple $FTS = (S, A, \rightarrow, I, \Pi, L)$, where

- S is the finite set of states,
- A is the finite set of actions,
- $\rightarrow \subseteq S \times A \times S$ is a transition relation,
- I is the set of initial states,
- Π is the set of atomic propositions and
- L is a labeling function which maps each state to the set of atomic propositions which are true on it.

Let $A(s) = \{a \in A : \exists s' \in S,\ s \rightarrow_a s'\}$ denote the set of active actions of s. Let $Post(s, a) = \{s' \in S : s \rightarrow_a s'\}$ denote the set of direct successors of s under action a. Denote $Post(s) = \bigcup_{a \in A(s)} Post(s, a)$ as the set of direct successors of s. A $path\ fragment$ is a finite or infinite sequence of states $s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$, where $s_{i+1} \in Post(s_i)$. A $path$ is a finite path fragment with a state having no direct successors or an infinite path fragment. The $trace$ of an infinite path fragment $s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$ is the sequence of the sets of atomic proposition corresponding to each state, i.e.,$L(s_0), L(s_1), \ldots, L(s_i), L(s_{i+1}), \ldots$. The language $\mathcal{L}(FTS)$ of FTS consists of all possible traces. A $execution$ of the FTS is a sequence $(s_0, a_0), (s_1, a_1), \ldots, (s_i, a_i), \ldots$ where $s_0 \in I, (s_i, a_i, s_{i+1}) \in \rightarrow$.

B. Linear temporal logic

Linear temporal logic (LTL) is a powerful formal language that specifies the desired system behaviors and the admissible environment behaviors, which was first used as a specification language by Pneuli [18]. By including temporal operators in reasoning propositions, LTL specifications are able to describe a wide range of properties of systems, such as safety, reachability, stability, recurrence, etc. An LTL formula consists of atomic propositions and logic operators. Different from proposition logic which uses the logic connectives: negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\Longrightarrow$), LTL incorporates temporal operators: next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$). Specifically, an LTL formula given a set of atomic propositions Π is defined inductively as follows:

- $True$ is an LTL formula,
- any atomic proposition $p \in \Pi$ is an LTL formula,
- given LTL formulas $\varphi,\ \varphi_1$ and φ_2, $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\bigcirc\varphi$ and $\varphi_1 \mathcal{U} \varphi_2$ are also LTL formula.

All other operators and all LTL formulas can be derived from these basic ones, i.e., $\varphi_1 \Longrightarrow \varphi_2 \Longleftrightarrow \neg\varphi_1 \wedge \varphi_2$, $\Box\varphi \Longleftrightarrow \neg\Diamond\neg\varphi$.

Definition 2.3: $LTL\ semantics$: a LTL formula is interpreted on an infinite sequence $\varepsilon = \varepsilon_0\varepsilon_1\varepsilon_2\varepsilon_3 \ldots$ where $\varepsilon_i \in 2^{\Pi}$. We say φ holds at position i, written as $\varepsilon_i \vDash \varphi$, if and only if (iff) φ holds for the remainder of the sequence ε starting from i. If $\varepsilon_0 \vDash \varphi$, we write $\varepsilon \vDash \varphi$.

The semantics of any LTL formula is defined as follows:

- $\varepsilon \vDash True$,
- for an atomic proposition $p \in \Pi$, $\varepsilon_i \vDash p$ iff $p \in \varepsilon_i$,
- $\varepsilon_i \vDash \neg\varphi$ iff $\varepsilon_i \nvDash \varphi$,
- $\varepsilon \vDash \varphi_1 \vee \varphi_2$ iff $\varepsilon \vDash \varphi_1$ or $\varepsilon \vDash \varphi_2$,
- $\varepsilon_i \vDash \bigcirc\varphi$ iff $\varepsilon_{i+1} \vDash \varphi$,
- $\varepsilon \vDash \varphi_1 \mathcal{U} \varphi_2$ iff there exists $j \geq i$ such that $\varepsilon_j \vDash \varphi$ and $\forall k \in [i, j), \varepsilon_k \vDash \varphi$.

Derived from the definition above, $\varepsilon_i \vDash \Box\varphi$ iff $\varepsilon_k \vDash \varphi, \forall k \in [i, +\infty)$. $\varepsilon \vDash \Diamond\varphi$ iff there exists $j \geq i$ such that $\varepsilon_j \vDash \varphi$. The set $LTL(\Pi)$ is all LTL formulas over Π.

Example 2.4: We can express various properties of interest in LTL formulas. A safety formula may be of the form $\Box(speed < 60\ miles)$ which states the requirement that the speed of a car is invariantly less than 60 miles throughout an execution. Safety formulas are supposed to ensure nothing bad happens. An example of guarantee formula would be $\Diamond(location = Exit)$ which means eventually the car will arrive at the exit at least once. This is an issue of reachability. A progress formula would be $\Box\Diamond(speed > 0)$ which asserts that the car has to move forward infinitely often during an execute since we can't forbid a car to brake but we don't allow that it never moves. In other words, we hope something good will always happen. The example of a response formula is $\Box(green\ light\ on \rightarrow \Diamond the\ i^{th}\ car\ moves\ forward)$, which describes how the i^{th} should react to the change of environment. A stability formula, i.e., $\Diamond\Box(0 < \Delta x < 100m)$, asserts that the property that the relative distance between each two adjacent cars of a platoon should always be in a range after a point of the execution. Some other properties can also be expressed in LTL language.

Given an FTS and a specification consisting a list of LTL formulas, synthesizing a control protocol is equivalent to finding a path that satisfies the specification, which is known as a model checking problem [16]. We refer the reader to [16] in which formal verification results can be found.

C. Reactive synthesis

For a system consisting of controlled and uncontrolled variables, we can treat the control protocol synthesis as a two-player game: the environment and the system. For example, the

brake of a vehicle is a controlled variable but the traffic light status is an uncontrolled variable. The two players alternatively acts with different purposes: the environment wants to falsify φ and the system wants to satisfy it. The system will make decisions based on its current state and the environment behavior. Generally, the reactive synthesis problem is to find a controller satisfy a specification of the following form:

$$\varphi = \varphi^e \rightarrow \varphi^s. \tag{1}$$

In other words, under some environment assumption specification, we are interested in whether there exists a controller that guarantee the system specification. Often this is also called Assumption/Guarantee (A/G) specification. The complexity of this synthesis problem is at most double exponential in the length of φ (more details can be found in [19]). We focus on a subset of LTL specifications, called Generalized Reactivity(1) (GR(1)) [20], which can be solved in polynomial time but at the same time still expressive. The GR(1) specification is in the following form:

$$\left(\varphi_{init} \wedge \Box\varphi^e \wedge \bigwedge_{i\in I_e} \Box\Diamond\varphi_i^e\right) \longrightarrow \left(\bigwedge_{i\in I_s} \Box\varphi_i^s \wedge \bigwedge_{i\in I_t} \Box\Diamond\varphi_i^s\right) \tag{2}$$

We shall convert the natural language specifications on the vehicle systems into LTL specifications in GR(1) form in the next section.

III. Specifications-based Platoon Control

In this section we introduce our framework of controlling the platoon in a regulated way. First we need to model the dynamics of a vehicle. The longitudinal motion of a vehicle can be modelled as a nonlinear system with parameters such as the mass of vehicle and the aerodynamic drag coefficient (see [21] for details). Without loss of generality, we use the following continuous linearized system to illustrate our framework:

$$\frac{d}{dt}\begin{bmatrix} s(t) \\ v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} s(t) \\ v(t) \\ a(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t), \tag{3}$$

where $x(t) = \begin{bmatrix} s(t) & v(t) & a(t) \end{bmatrix}^T \in \mathcal{X} \subseteq \mathbb{R}^3$ is the state vector, s is the relative distance between the roadside reference point and the vehicle, v is the velocity of the vehicle, and a is the acceleration. $u \in \mathcal{U} \subseteq \mathbb{R}$ is the control input. $\mathcal{X}$ is the continuous state space and $\mathcal{U}$ is the set of admissible control input. Since the continuous states are infinite in which case the model-checking based synthesis cannot be used, the next step is to build a finite transition system (FTS) abstracting from the original system. Roughly speaking, the continuous state space is partitioned into many proposition preserving cells which serves as states in FTS. The transition between cells is defined if and only if there exists a continuous control input taking the system from any point in the source cell to some point in the destination cell. Once we have a discrete controller synthesized based on the FTS and some given specifications which will be discussed later, we can $translate$ a sequence of abstract discrete states back to a sequence of continuous states [15]. A lot of research results and examples on abstraction approach can be found in [14], [22]–[24]. Note that we do not add any disturbance but we consider the deterministic system whose behavior is totally controlled by the input. The extension is not that difficult. For example, the disturbance can be considered by the continuous controller which generates a control input taking the system to the destination against any possible disturbance. Assume the FTS for a vehicle i is $\mathcal{V}_i = (S, A, \rightarrow, I, \Pi, L)$ and all the vehicles are identical.

The dynamics information of a vehicle can be accessed by another vehicle via two approaches. One is from the distance measurement sensor, i.e., radar or lidar sensors. The follower can detect the relative distance between the leader in front of it and itself but obtain no other information like velocity or acceleration, though it can estimate these information (delayed and inaccurate) based on the sensor measurements and its own dynamics information. The other way of information flow resorts to the wireless communication like DSRC [25], [26], which can share all the dynamic information of one vehicle to its neighbors. The disadvantage is that the data packets may drop anytime and the packet drop rate increases with the relative distance and the channel load, contrast to stability of collecting sensor measurements. In our case, we consider a vehicle only communicates with its preceding vehicle and its follower with non-deterministic packet loss.

We consider a platoon of $N+1$ vehicles which has a leader C_0 and N followers. The followers are identified by $C_i, i \in G = [1, N]$ with FTS=$\mathcal{V}_i$. The identifier i is smaller if C_i is closer to C_0. The examples of an atomic proposition could be $\{x_i - x_{i-1} > 5\}, \{v_i \leq 20\}, \{a_i < 5\}$, where x_i is the distance between C_i and a given roadside reference point and the unit is m, v_i is the velocity of C_i and the unit is m/s, a_i is the acceleration of C_i and the unit is m/s^2. Each AP is mapped to a set of discrete states in $\mathcal{V}_i$. A boolean environment variable c_i acts as the indicator that whether C_i is receiving data packets from C_{i-1}. The assumption on c_i, expressed in LTL formula, is

$$\varphi_e = \bigwedge_{i\in G} \varphi_e^i, \tag{4}$$

$$\varphi_e^i = \Box\Diamond c_i. \tag{5}$$

Now we write a list of example specifications in the form of LTL formulas to express the desired behaviors of the platoon. The following specifications can be modified or extended for different purposes.

1) Safety specification $\varphi_s = \bigwedge_{i\in G} \varphi_{s,1}^i \wedge \bigwedge_{i\in G} \varphi_{s,2}^i$: we have

$$\varphi_{s,1}^i = \Box(x_i - x_{i-1} > 0) \tag{6}$$

for collision avoidance, and

$$\varphi_{s,2}^i = \Box(0 \leq v_i \leq 20) \tag{7}$$

for setting speed limit and no backing off.

2) Progress specification $\varphi_p = \bigwedge_{i\in G} \varphi_{p,1}^i \wedge \bigwedge_{i\in G} \varphi_{p,2}^i$: we have

$$\varphi_{p,1}^i = \Box\Diamond(v_i > 0) \tag{8}$$

because the vehicle is not allowed to stop forever.

$$\varphi^i_{p,2} = \bigwedge_{i\in G} \square\diamond(x_i - x_{i-1} \leq 5) \tag{9}$$

makes the follower will always eventually approach its preceding vehicle within 5 meters to form a platoon.

3) Response specification φ_r: $\varphi_r = \bigwedge_{i\in G} \varphi^i_{r,1} \wedge \bigwedge_{i\in G} \varphi^i_{r,2}$: We list two simple response specifications here. $\varphi_{r,1}$ means when C_i receives the dynamics information of C_{i-1}, it can predict the next position of C_{i-1} and eventually get closer while avoid collision. When there is no communication, C_i should take a conservative action instead.

$$\varphi^i_{r,1} = \square\diamond(c \rightarrow (x_i - \bigcirc x_{i-1}) > (\bigcirc x_i - \bigcirc x_{i-1})) \tag{10}$$

$$\varphi^i_{r,2} = \square(\neg c \rightarrow (x_i - \bigcirc x_{i-1}) \leq (\bigcirc x_i - \bigcirc x_{i-1})) \tag{11}$$

4) Comfortability specification φ_c: φ_c: $\varphi_r = \bigwedge_{i\in G} \varphi^i_{c,1} \wedge \bigwedge_{i\in G} \varphi^i_{c,2}$:
To make the passengers comfortable, we set some limit on the change of acceleration and deceleration. When the relative distance is large, the vehicle should smoothly accelerate and decelerate.

$$\varphi^i_{c,1} = \square(-5 \leq a_i \leq 5) \tag{12}$$

$$\varphi^i_{c,2} = \square\left((x_i - x_{i-1} > 20) \rightarrow (|a_i - \bigcirc a_i| \leq 2)\right) \tag{13}$$

While he leader of the platoon is usually driven by a driver or separately controlled from the followers by the central controller, we still impose some specifications that constrain its behavior. For example, if the leader stands still or drives too fast, the followers are not able to satisfy their specifications. We set the specification for the leader as follows

$$\varphi^l = \varphi^0_{s,2} \wedge \varphi^0_{p,1} \wedge \varphi^0_{c,1}. \tag{14}$$

The specification for all the followers is

$$\varphi^f = \varphi_s \wedge \varphi_p \wedge \varphi_r \wedge \varphi_c. \tag{15}$$

Distributed controller synthesis problem : Given the FTS for each vehicle and the global Assumption/Guarantee specification

$$\varphi = \left(\varphi_e \wedge \varphi^l\right) \rightarrow \varphi^f, \tag{16}$$

one needs to automatically synthesize a sequence of control signals, also called control protocol,

$$u^1_i, u^2_i, \ldots \in \mathcal{U}$$

for each subsystem i starting from any initial states to satisfy the specification for any sequence of environment status. If there exists at least a sequence of control input to take the given reactive system to satisfy the specification, then the specification is said to be *synthesizable*.

IV. Distributed Controller Synthesis

We solve the distributed synthesis problem by dividing the platoon into N subsystems. Each subsystem $\mathcal{M}_i = (i-1, i), i \in G$ consists of two vehicles. The FTS C_{i-1} together with the environment variable c_i acts as the environment for C_i. The next step is to compose an appropriate A/G specification φ^i_{sub} for $\mathcal{M}_i$ and synthesize a distributed controller for C_i such that the whole platoon satisfies φ. A natural idea is to decompose the global specification into

$$\begin{aligned}\varphi^i_{sub} = &\left(\varphi^i_e \wedge \varphi^{i-1}_{s,2} \wedge \varphi^{i-1}_{p,1} \wedge \varphi^{i-1}_{c,1} \wedge \varphi^{i-1}_{c,2}\right) \\ &\rightarrow \left(\varphi^i_{s,1} \wedge \varphi^i_{s,2} \wedge \varphi^i_{p,1} \wedge \varphi^i_{c,1} \wedge \varphi^i_{c,2}\right).\end{aligned} \tag{17}$$

In other words, if assuming C_{i+1} satisfies the specification on its own properties such as $x_{i-1}, v_{i-1}, a_{i-1}$ and the environment specification holds such that there exists a controller that guarantees C_i to satisfy the specification on its own properties and relative distance, we can prove that the global A/G specification for the whole platoon can be satisfied by induction. However, in general φ^i_{sub} can't be decomposed from the global specification φ directly. A counterexample is as follows:

Example 4.1: Suppose a simple global specification is

$$\bigwedge_{i\in G} \varphi^i_{s,2} \wedge \bigwedge_{i\in G} \varphi^i_{p,2}.$$

If we directly filter the relevant specification for $\mathcal{M}_i$ out, say, the environment assumption for C_{i-1} is

$$\varphi^i_{sub,e} := \varphi^{i-1}_{s,2}. \tag{18}$$

and the system guarantee for C_i is

$$\varphi^i_{sub,s} := \varphi^i_{s,2} \wedge \varphi^i_{p,2}. \tag{19}$$

Thus the A/G specification for $\mathcal{M}_i$ is

$$\varphi^i_{sub} := \varphi^i_{sub,e} \rightarrow \varphi^i_{sub,s}.$$

If there exists a controller such that the sequence of states of C_i satisfy φ^i_{sub} for any $i \in G$. The global specification can be satisfied automatically. However, the controller does not exist. If the first m cars in the platoon are always at the speed of 20 m/s and the distance of each one to their neighbors is 5 m, the $(m+1)$st car is left 10 m behind the mth car. There is no controller for C_{m+1} to catch up with C_m satisfying the safety specification and progress specification simultaneously.

Since the global specification decomposition may not work due to the lack of a feasible controller like the example above, one has to find a better approach for decomposition in which the environment assumption in each subsystem will be relaxed. One way of relaxing the environment assumption for $\mathcal{M}_i$ is to restrict the behavior of vehicle in the guarantee specification for $\mathcal{M}_{i-1}$. In the next section, we introduce the concept of invariant specification by which we can synthesize a controller for each vehicle to satisfy the guarantee specification while slightly relaxing the assumption on the leading vehicle.

A. Invariant Specification

We denote $\varphi^i_{private}$ as an LTL formula containing only atomic propositions of x_i, v_i, a_i and denote φ^i_{public} as an LTL formula only containing the relative distance $x_i - x_{i-1}$, velocity $(v_i - v_{i-1})$ and acceleration $(a_i - a_{i-1})$. Then the A/G specification in (17) can be written into

$$\left(\varphi_e^{i-1} \wedge \varphi_{private}^{i-1}\right) \rightarrow \left(\varphi_{private}^{i} \wedge \varphi_{public}^{i}\right) \tag{20}$$

If we can find a φ^i_{new} such that there exists a controller for any $C_i, i \in N$ satisfy the following specification,

$$\begin{aligned}\left(\varphi_e^{i-1} \wedge \varphi_{private}^{i-1} \wedge \varphi_{new}^{i-1}\right) \rightarrow \\ \left(\varphi_{private}^{i} \wedge \varphi_{new}^{i} \wedge \varphi_{public}^{i}\right).\end{aligned} \tag{21}$$

Then we define

$$\varphi^i_{inv} = \varphi^i_{private} \wedge \varphi_i \tag{22}$$

as an $invariant\ specification$ for C_i.

Example 4.2: In previous example, we can define

$$\varphi^i_{inv} = \varphi^i_{s,2} \wedge \square\diamond(v_i \leq 20) \tag{23}$$

as the invariant specification. Thus the new A/G specification for $\mathcal{M}_i$ is

$$\varphi^i_{sub,new} = \varphi^{i-1}_{inv} \rightarrow \left(\varphi^i_{inv} \wedge \varphi^i_{p,2}\right). \tag{24}$$

We can use TuLiP to synthesize a controller for this simple system.

We give a definition on the partial order over the set of all LTL specifications.

Definition 4.3: $Partial\ order$: Let $\varphi_1, \varphi_2 \in LTL(\Pi)$, then $\varphi_1 \preceq \varphi_2$ iff $\mathcal{L}(\varphi_1) \subseteq \mathcal{L}(\varphi_2)$.

Before we present the main theorem, we need the following lemma on boolean operation.

Lemma 4.4: If $a \wedge b \rightarrow c \wedge d$ and $c \wedge e \rightarrow f$, then $a \wedge b \wedge e \rightarrow c \wedge d \wedge f$.

Proof: If $a \wedge b \wedge e \rightarrow c \wedge d \wedge f$ is $False$, we have

$$a \wedge b \wedge e \text{ is } True \tag{25}$$

and

$$c \wedge d \wedge f \text{ is } False. \tag{26}$$

If we assume $a \wedge b \rightarrow c \wedge d$ is $True$, from (25) we have $a \wedge b$ is $True$ and $c \wedge d$ is $True$. Thus h is $False$ from (26). Since c and e are $True$, $c \wedge e \rightarrow f$ is $False$. Therefore, $a \wedge b \rightarrow c \wedge d$ and $c \wedge e \rightarrow f$ cannot hold simultaneously. This completes proof. ∎

Now we state a theorem that if we can find a proper invariant specification, we can turn the distributed controller synthesis problem into a set of solvable two-player games.

Theorem 4.5: Assume that the communication channel specification φ_e and the specification of the leading vehicle of the platoon φ^i_{inv} hold. If there exists an invariant specification

$$\varphi^i_{inv} \preceq \varphi^i_{private}$$

such that there is a controller for $\mathcal{M}_i$ satisfying the specification in the following form:

$$\left(\varphi_e^i \wedge \varphi_{inv}^{i-1}\right) \rightarrow \left(\varphi_{inv}^{i} \wedge \varphi_{public}^{i}\right) \tag{27}$$

for any $i \in G$, then φ^f in (16) can be guaranteed. In other words,

$$\left(\varphi_e \wedge \varphi^0_{inv}\right) \rightarrow \varphi^f \tag{28}$$

is synthesizable.

Proof: Note that the control protocol satisfying an A/G specification only ensure the guarantee specification only when the environment restricts its behavior under its assumption. Otherwise, we cannot say that the guarantee part holds even if the A/G specification is synthesizable and satisfied by some controllers.

Given that the leader of the platoon C_0 satisfies the assumption specification φ^0_{inv} and the environment assumption φ^1_e holds, the specification in (27) for $\mathcal{M}_1$ is synthesizable, i.e., the automatically synthesized controller can guarantee that $\varphi^1_{inv} \wedge \varphi^1_{public}$. Since

$$\left(\varphi^2_e \wedge \varphi^1_{inv}\right) \rightarrow \left(\varphi^2_{inv} \wedge \varphi^2_{public}\right) \tag{29}$$

is synthesizable, from Lemma 4.4 we have the synthesizable A/G specification

$$\left(\varphi^2_e \wedge \varphi^1_e \wedge \varphi^0_{inv}\right) \rightarrow \left(\varphi^2_{inv} \wedge \varphi^2_{public} \wedge \varphi^1_{inv} \wedge \varphi^1_{public}\right). \tag{30}$$

Assuming that

$$\bigwedge_{m=1}^{i} \varphi^m_e \wedge \varphi^0_{inv} \rightarrow \bigwedge_{m=1}^{i} \varphi^m_{inv} \wedge \bigwedge_{m=1}^{i} \varphi^m_{public} \tag{31}$$

is synthesizable. Based on Lemma 4.4 and (27), we have the synthesizable specification

$$\bigwedge_{m=1}^{i+1} \varphi^m_e \wedge \varphi^0_{inv} \rightarrow \bigwedge_{m=1}^{i+1} \varphi^m_{inv} \wedge \bigwedge_{m=1}^{i+1} \varphi^m_{public}. \tag{32}$$

Therefore, we can inductively conclude that there exists a control protocol for each subsystem $\mathcal{M}_i, i \in G$ such that

$$\left(\varphi_e \wedge \varphi^0_{inv}\right) \rightarrow \bigwedge_{i \in G} \left(\varphi^i_{inv} \wedge \varphi^i_{public}\right) \tag{33}$$

Since $\varphi^i_{inv} \preceq \varphi^i_{private}$, the modified guarantee specification is a sufficient condition that the original guarantee specification in (16) can be ensured by some control protocols, i.e.,

$$\left(\varphi_e \wedge \varphi^0_{inv}\right) \rightarrow \varphi^f \tag{34}$$

∎

Note that the only difference between (16) and (34) is that we relax the assumption on the leader by restricting its behavior described by φ^0_{inv}.

Remark 4.6: Searching a feasible invariant specification falls in the category of the minimal revision problem for specifications, say, revising or relaxing the specifications such that they are as *close* as the initial user intent, like [27] [28]. Basically, one may easily find $\varphi^i_{inv} \preceq \varphi^i_{private}$ by relaxing $\varphi^i_{private}$. The key issue is how to revise the specifications minimally and automatically, which is left as future work.

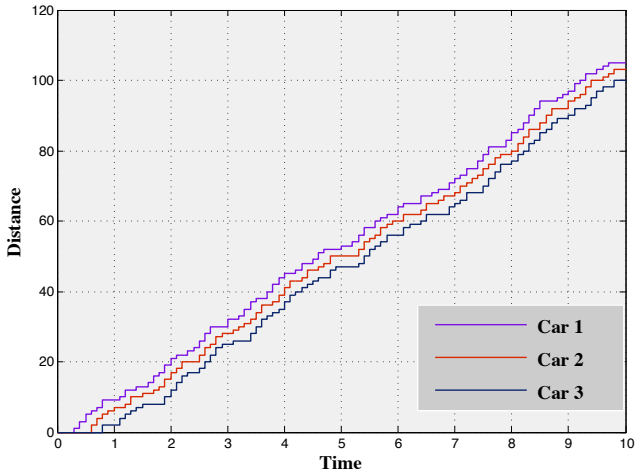

Fig. 1. Example of a platoon of three cars. Car 1 is the leader which satisfies (14) and the invariant specification in (23).

V. Examples

We use a simple example to illustrate the synthesis process. The platoon consists of three cars, i.e., one leader and two followers. We directly use the simplified discrete transition system to describe the dynamics of the vehicle. The number of system states is 48, i.e., 6 states for the relative distance, 4 states for the velocity and 2 states for the communication channel. The specification of acceleration is guaranteed within the setting. The controller is automatically synthesized satisfying (24) by TuLiP, which outputs a script that generates a Mealy machine in Simulink. In Fig. 1 we show a realization of the positions of three cars in the platoon vs time (sampling time is $0.1s$). Note that the controller for the (Car 1, Car 2) pair and (Car 2, Car 3) pair is the same and we only need to synthesize one common controller for the whole platoon.

VI. Concluding Remarks

In this paper we formulate the distributed control of automated vehicles problem using LTL specifications. Thanks to the expressive power of LTL, we can translate the behavioral requirements of the platoon in natural language into a set of LTL-based specifications. Generally the distribution of the global specification is hard to solve. By introducing the invariant specification, we divide the global specification into a number of GR(1) specifications for each reactive system. The satisfiability of each specification works together to guarantee the satisfiability of the global specifications. Then we can automatically synthesize a distributed controller for each vehicle in the platoon. Future work includes investigation of automatically search for the invariant specification even for the solution set and extension for lateral control.

References

[1] P. Varaiya, "Smart cars on smart roads: problems of control," *IEEE Trans. Autom. Control*, vol. 38, no. 2, pp. 195–207, 1993.

[2] R. Horowitz and P. Varaiya, "Control design of an automated highway system," *Proc. IEEE*, vol. 88, no. 7, pp. 913–925, 2000.

[3] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in dsrc," in *Proc. ACM Int. Workshop on Vehicular ad hoc networks*, 2004, pp. 19–28.

[4] S. Sheikholeslam and C. A. Desoer, "Longitudinal control of a platoon of vehicles," in *Proc. American Control Conf., 1990*, 1990, pp. 291–296.

[5] ——, "Longitudinal control of a platoon of vehicles with no communication of lead vehicle information: a system level study," *IEEE Trans. Vehicular Technology*, vol. 42, no. 4, pp. 546–554, 1993.

[6] D. N. Godbole and J. Lygeros, "Longitudinal control of the lead car of a platoon," *IEEE Trans. Vehicular Technology*, vol. 43, no. 4, pp. 1125–1135, 1994.

[7] S. S. Stankovic, M. J. Stanojevic, and D. D. Siljak, "Decentralized overlapping control of a platoon of vehicles," *IEEE Trans. Control Systems Technology*, vol. 8, no. 5, pp. 816–832, 2000.

[8] D. Swaroop, J. K. Hedrick, and S. Choi, "Direct adaptive longitudinal control of vehicle platoons," *IEEE Trans. Vehicular Technology*, vol. 50, no. 1, pp. 150–161, 2001.

[9] G. Guo and W. Yue, "Autonomous platoon control allowing range-limited sensors," *IEEE Trans. Vehicular Technology*, vol. 61, no. 7, pp. 2901–2912, 2012.

[10] S.-Y. Han, Y.-H. Chen, L. Wang, and A. Abraham, "Decentralized longitudinal tracking control for cooperative adaptive cruise control systems in a platoon," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, 2013.

[11] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2005, pp. 2020–2025.

[12] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, 2008.

[13] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proc. ACM Int. Conf. Hybrid systems: computation and control*, 2010, pp. 101–110.

[14] ——, "Synthesis of control protocols for autonomous systems," *Unmanned Systems*, vol. 1, no. 01, pp. 21–39, 2013.

[15] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[16] C. Baier, J.-P. Katoen *et al.*, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[17] A. Pneuli and R. Rosner, "Distributed reactive systems are hard to synthesize," in *Proc. Symp. on Foundations of Computer Science*. IEEE, 1990, pp. 746–757.

[18] A. Pnueli, "The temporal logic of programs," in *Proc. IEEE 18th Annual Symp. Foundations of Computer Science*, 1977, pp. 46–57.

[19] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proc. Symp. Principles of programming languages*, 1989, pp. 179–190.

[20] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[21] P. A. Ioannou and C.-C. C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. Vehicular Technology*, vol. 42, no. 4, pp. 657–672, 1993.

[22] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[23] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, no. 2, pp. 566–571, 2009.

[24] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[25] F. Bai and H. Krishnan, "Reliability analysis of dsrc wireless communication for vehicle safety applications," in *Proc. IEEE Intelligent Transportation Systems Conf.*, 2006, pp. 355–362.

[26] L. Cheng, B. E. Henty, D. D. Stancil, F. Bai, and P. Mudalige, "Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 ghz dedicated short range communication (dsrc) frequency band," *IEEE J. Selected Areas in Communications*, vol. 25, no. 8, pp. 1501–1516, 2007.

[27] K. Kim, G. E. Fainekos, and S. Sankaranarayanan, "On the revision problem of specification automata," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2012, pp. 5171–5176.

[28] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2011, pp. 40–45.