2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

Ensuring Safety at Intelligent Intersections: Temporal Logic Meets Reachability Analysis

Kaj Munhoz Arfvidsson*, Frank J. Jiang*, Karl H. Johansson, Jonas Mårtensson

***Abstract*— In this work, we propose an approach for ensuring the safety of vehicles passing through an intelligent intersection. There are many proposals for the design of intelligent intersections that introduce central decision-makers to intersections for enhancing the efficiency and safety of the vehicles. To guarantee the safety of such designs, we develop a safety framework for intersections based on temporal logic and reachability analysis. We start by specifying the required behavior for all the vehicles that need to pass through the intersection as linear temporal logic formula. Then, using temporal logic trees, we break down the linear temporal logic specification into a series of Hamilton-Jacobi reachability analyses in an automated fashion. By successfully constructing the temporal logic tree through reachability analysis, we verify the feasibility of the intersection specification. By taking this approach, we enable a safety framework that is able to automatically provide safety guarantees on new intersection behavior specifications. To evaluate our approach, we implement the framework on a simulated T-intersection, where we show that we can check and guarantee the safety of vehicles with potentially conflicting paths.**

I. Introduction

In recent years, the need for intelligent intersections in the transportation network has become increasingly evident. Since intersections are often both inefficient and dangerous [1], there is a significant amount of work that has gone into proposing updates to traditional intersection management techniques that involve higher levels of autonomy and vehicle-to-infrastructure (V2I) communication, e.g. [2], [3]. While there is a significant emphasis on safety for the design of intelligent intersections, there is still little consensus around how we should provide safety guarantees that are flexible to possible changes in the intersection specification. One of the core challenges is the difficulty in computing the maximal controlled invariant sets for intersections in a general, computationally-tractable way, since finding the exact solution is an NP-complete problem [4]. To address this, several approaches propose approximate solutions to the problems where the maximal controlled invariant set is conservatively approximated and leverage assumptions made about how the vehicles will pass through the intersection [4]–[6]. Alternatively, some take a probabilistic approach and provide lower bounds on vehicle collision probabilities in intelligent intersections [7]. While these approaches do provide safety guarantees, they are built upon specific intersection traffic rules. Due to the diverse and evolving requirements of traffic passing through intersections, there is interest to further investigate approaches that are able to provide more flexible safety guarantees that easily adapt to updates to changing intersection traffic rules.

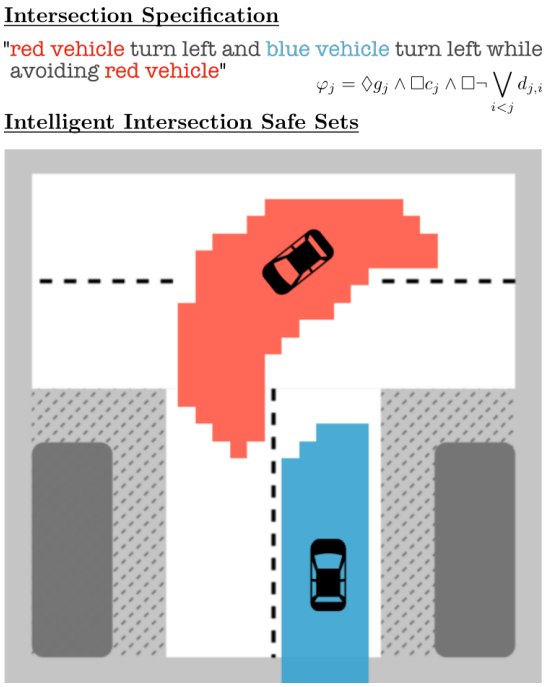

Fig. 1. T-Intersection example with two vehicles passing through and the safe sets computed for the intersection specification.

For more flexible safety guarantees in intelligent transportation systems, researchers have recently proposed several approaches based on the formalization of traffic rules. Used in the specification and verification of various types of complex systems [8], temporal logic offers a compelling approach for formalizing requirements on systems in a way that is both flexible and approachable to human designers. For example, [9] shows that they can formalize current German intersection traffic rules using metric temporal logic. Specifically for designing intersection management, [10] use linear temporal logic to specify and verify the safety of

*Indicates equal contribution

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems, and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. It was also partially supported by the Swedish Research Council, Swedish Research Council Distinguished Professor Grant 2017-01078, the Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant, and the Swedish Innovation agency (Vinnova), under grant 2021-02555 Future 5G Ride, within the Strategic Vehicle Research and Innovation program (FFI).

All authors are with the Division of Decision and Control Systems, EECS, KTH Royal Institute of Technology, Malvinas väg 10, 10044 Stockholm, Sweden {kajarf, frankji, kallej, jonas1}@kth.se. They are also affiliated with the Integrated Transport Research Lab and Digital Futures.

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588818

an intersection management algorithm. While they do not use temporal logic, [11] similarly develop formal specifications for intersection management by formalizing the responsibility-sensitive safety model [12] using Hoare Logic that can be used for discovering conditions that guarantee safety of the intersection. In this work, we show how to take intersection rules that are formalized in linear temporal logic and leverage temporal logic trees [13] to directly use Hamilton-Jacobi (HJ) reachability analysis to verify the feasibility of the intersection rules.

The main contribution of this paper is a safety framework that verifies the feasibility of a multi-vehicle specification in an intelligent intersection. Specifically, the contributions of the paper can be summarized as follows:

1) we present a linear temporal logic-based sequential path planning approach for intelligent intersections,
2) we detail the construction of temporal logic trees for verifying the feasibility of the sequential path planning,
3) we evaluate the approach by verifying the safe crossing of vehicles through a T-intersection. The code used for the evaluation is publicly available[1].

Taking inspiration from the sequential path planning approaches developed for aerial vehicles in [14], [15], the contributed approach starts with formalizing an intersection specification with linear temporal logic formulae. Then, by using temporal logic trees for the verification of these formulae, we develop an approach that results in both safety guarantees for the intelligent intersection and is also able to automatically handle any changes to the specifications. Through the use of HJ reachability analysis, the approach is also able to handle the nonlinearities of road vehicle models and the complex environments of intersections. Moreover, in our numerical evaluation, we find preliminary indications that, by using modern software libraries, the online verification computation time is potentially fast enough to be implemented on real intelligent management systems.

The remainder of the paper is organized as follows. In Section II, we provide the necessary preliminary material for the presented approach and use a motivating example to state the problem addressed in this work. In Section III, we describe our approach to verifying the feasibility and safety of temporal logic specifications for intelligent intersections. In Section IV, we numerically evaluate the approach on a three vehicle T-intersection scenario. In Section V, we conclude the paper with a discussion about our work and future directions.

II. Problem Formulation

In this section, we introduce the necessary preliminary material for our approach. We focus on introducing the notation and definitions that are key for the application of temporal logic and reachability analysis to ensuring safety at intelligent intersections. Finally, we state the specific problem addressed in this work.

[1]https://github.com/kaarmu/safe_intersections

A. Multi-Vehicle Model

In this section, we define the multi-vehicle model we use for ensuring the safety of intelligent intersections. We start by defining the vehicle model for a single vehicle and then collect the single vehicle models into a collective multi-vehicle model.

For a single vehicle i, let $z_i = [x_i, y_i, \theta_i, \delta_i, v_i]^\top$ be the state, where x_i, y_i, θ_i, δ_i, and v_i are the vehicle's x-position, y-position, heading angle, steering angle, and velocity, respectively. Then, let $u_i = [s_i, a_i]^\top$ be the input, where s and a are the steering rate and acceleration inputs into the vehicle, respectively. Explicitly, we model the dynamics with

$$\dot{z}_i = f_i(z_i) + g_i(z_i)u_i, \tag{1}$$

where

$$f_i(z_i) = \begin{bmatrix} v_i \cos\theta_i \\ v_i \sin\theta_i \\ \frac{v_i \tan\delta_i}{L_i} \\ 0 \\ 0 \end{bmatrix}, \; g_i(z_i) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

L_i is the wheel-base length of the vehicle. Here, $z_i \in \mathbb{R}^5$ and $u_i \in \mathbb{U} \subset \mathbb{R}^2$, where $\mathbb{U}$ is the full set of physically feasible steering rates and accelerations/decelerations we assume vehicles passing through the intersection can implement. To analyze the possible decisions vehicles could make while passing through the intersection, we let $u_i(\cdot) \in U$ be vehicle i's physically feasible control policy, where U is the function space containing all physically feasible vehicle control policies. We also denote a trajectory of vehicle i with $\zeta_i(\cdot; z_{i,0}, t_0, u_i(\cdot))$, which is the trajectory starting from initial state $z_{i,0} = \zeta_i(t_0; z_{i,0}, t_0, u_i(\cdot))$ under control policy $u_i(\cdot)$. For simplicity, we sometimes describe the trajectory of vehicle i with $\zeta_i(\cdot)$.

For analyzing the behavior of multiple vehicles in an intersection, we also define a multi-vehicle model. For an intersection with N vehicles, let the full multi-vehicle state and control input be $z = [z_1, z_2, \ldots, z_N]^\top$ and $u = [u_1, u_2, \ldots, u_N]^\top$, respectively. Then, we can write the full multi-vehicle dynamics as the following:

$$\dot{z} = f(z) + g(z)u, \tag{2}$$

Finally, we write the collective trajectory of the multi-vehicle system as $\zeta(\cdot; z_0, t_0, u(\cdot))$, which we will also sometimes write as $\zeta(\cdot)$ for simplicity.

For the remainder of this paper, we will work with time-state sets of of the multi-vehicle system. First, denote the full state space of the multi-vehicle system as $\mathbb{S}$. We denote time-state sets for the multi-vehicle system as $\mathcal{S} \subseteq \mathbb{S} \times \mathbb{R}$. Then, for retrieving state sets at particular times we define the time-state set map $\Omega : \mathbb{R} \to \mathbb{S}$. An $\mathcal{S}$ has a corresponding $\Omega_\mathcal{S}$ where

$$\mathcal{S} = \bigcup_{t \in \mathbb{R}} \{(z, t) \mid z \in \Omega_\mathcal{S}(t)\}. \tag{3}$$

If, however, $\{(z, t) \in \mathcal{S} \mid \exists t' \neq t,\, z \notin \Omega_\mathcal{S}(t')\}$ is non-empty then we refer to $\mathcal{S}$ as being invariant all time.

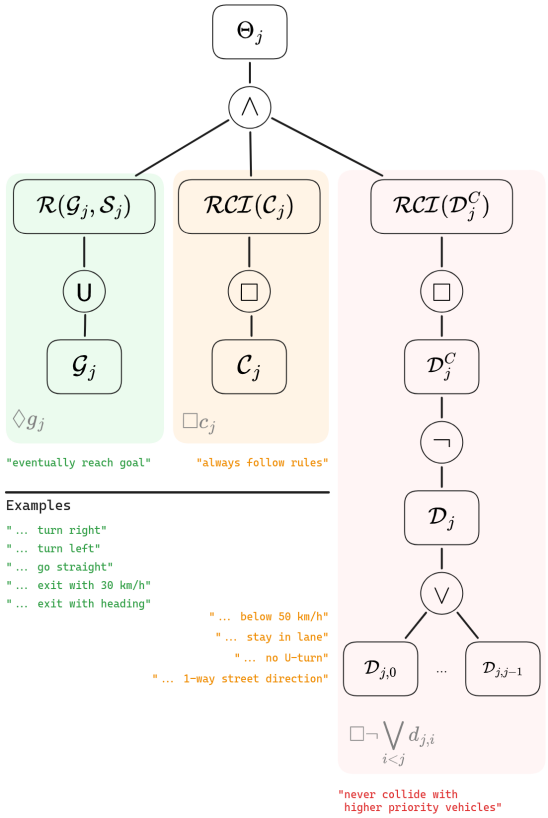

Fig. 2. Illustration of the constructed TLT for the jth vehicle's individual LTL specification (5) that ensures safety at intersections.

B. Temporal Logic

We define requirements for the intelligent intersection using temporal logic. This allows us to create high-level, human-readable requirements on how vehicles should be moving through the intersection. For example, in Fig. 1, we might specify the simple requirements "vehicle should turn left and avoid collisions with other vehicles". Using temporal logic, one can write a formal equation representing this requirement using operators that correspond to intuitive concepts in human language, i.e. "always stay below 50 km/h", "always stay in lane", "eventually turn left", or "eventually exit intersection with 30 km/h" (more examples listed in Fig. 2). Others have leveraged the intuitive and rich specification capability of temporal logic in a variety of transportation problems [9], [16], [17]. In this work, we work with Linear Temporal Logic (LTL) since it yields the benefits of temporal logic, while being simple to work with and understand. Specifically, we use the operators $\{\neg, \vee, \wedge, \mathsf{U}, \Diamond, \Box\}$, which correspond to the Boolean operators "not", "or", "and", and the temporal operators "until", "eventually", and "always", respectively. We note that similar methods to the one we present in this work can be applied to Signal Temporal Logic by adapting the work with the approach presented in [18]. We direct readers interested in seeing the same LTL syntax used in this work to [19].

C. Temporal Logic Tree

Once we have specified an LTL specification for our multi-vehicle model, we need to check the feasibility of the specification. In this work, we will perform these computations using a computational model called Temporal Logic Trees [13] (example illustrated in Fig. 2). Intuitively speaking, the leaf nodes of the temporal logic tree are the goals of the multi-vehicle system. From the goals, we perform a series of reachability analyses to find the joint set of feasible trajectories that satisfy the intersection specification. When the computation finishes, if the TLT has been successfully constructed, we know the intersection specification is feasible and possible to satisfy. This verification result is detailed in [13, Theorem V.1]. To construct temporal logic trees, we need to compute two kinds of reachable sets which are defined below.

Definition 2.1: **(Backward Reachable Tube)** Given the full multi-vehicle system (2), a computation time horizon T, a constraint time-state set $\mathcal{C} \subseteq \mathbb{S} \times \mathbb{R}$, and a target time-state set $\mathcal{G} \subseteq \mathbb{S} \times \mathbb{R}$, we define the backward reachable tube as

$$\begin{aligned}\mathcal{R}(\mathcal{G};\mathcal{C}) = \{(z,t) \mid \exists u(\cdot) \in U,\\ \exists \tau \in [t,T),\ \zeta(\tau; z, t, u(\cdot)) \in \Omega_{\mathcal{G}}(\tau),\\ \forall \tau' \in [t,\tau),\ \zeta(\tau'; z, t, u(\cdot)) \in \Omega_{\mathcal{C}}(\tau')\},\end{aligned}$$

where $\mathcal{R}(\mathcal{G};\mathcal{C})$ contains the set of states that are able to reach the target set $\mathcal{G}$ while respecting the constraint set $\mathcal{C}$.

Definition 2.2: **(Robust Control Invariant Set)** For the full multi-vehicle system (2), computation time horizon T, and constraint set $\mathcal{C} \subseteq \mathbb{S} \times \mathbb{R}$, $\mathcal{RCI}(\mathcal{C}) \subseteq \mathbb{S} \times \mathbb{R}$ the largest robust control invariant set such that $\forall (z,t) \in \mathcal{RCI}(\mathcal{C})$ there $\exists u(\cdot) \in U$ such that $\forall \tau \in [t,T]$, $\zeta(\tau; z, t, u(\cdot)) \in \mathcal{C}$.

By using $\mathcal{R}(\cdot)$ and $\mathcal{RCI}(\cdot)$, we are able to fully construct temporal logic trees. For more details, we refer readers to [13].

D. Problem Statement

In this work, we are interested in developing an approach for guaranteeing that vehicles are safely coordinated through intelligent intersections. For example, in Figure 1, we illustrate a T-intersection example that we will evaluate using our approach. The two vehicles are approaching the T-intersection, vehicle $\mathcal{V}_1$ from the right and vehicle $\mathcal{V}_2$ from below. Both intend to make a left turn, posing a potential risk of collision. For each of these vehicles, we can specify their overall behavior using the LTL formulae:

$$\varphi_1 = \varphi_{\text{turn left}} \wedge \varphi_{\text{safety}},\ \varphi_2 = \varphi_{\text{turn left}} \wedge \varphi_{\text{safety}},$$

where φ_1 is $\mathcal{V}_1$'s specification and φ_2 is $\mathcal{V}_2$'s specification. These LTL formulae reflect each vehicle's individual specification of turning left while staying safe. Then, the specification for the intelligent intersection can be described by the following multi-vehicle LTL formula:

$$\varphi = \varphi_1 \wedge \varphi_2. \tag{4}$$

This simple example is representative of the primary safety challenge of intersections: how can vehicles pass through

the intersection while avoiding collisions? For the rest of the work, we will refer to this challenge as the "intersection safety challenge." As was mentioned earlier, this challenge is addressed and solved by previous works. However, many of these solutions provide complete solutions that may be difficult to safely extend or adapt, as the safety guarantees are often built on the particular design decisions in the intersection management algorithm. Thus, much like [9], [10], we seek to leverage the richness of temporal logic to develop a safety framework that can solve for solutions to the intersection safety challenge in a way that can be easily adapted and built upon. Explicitly, given a multi-vehicle specification for an intersection, such as (4), we seek to automatically verify its feasibility to guarantee the full specification is satisfied, while considering all the vehicle's dynamics and decision uncertainty.

III. Safety Verification for Intersections

In this section, we develop our approach to finding solutions to the intersection safety challenge described in Section II-D. We start by posing the intersection safety challenge as a sequential path planning problem, inspired by the method developed in [14]. Then, we formalize the sequential path planning problem into LTL formulae. After we obtain LTL formulae, we detail how to use TLT to compute satisfaction sets for the sequential path planning specification. To make the approach more practical, we detail the computational approaches we employ to construct the TLT with a reasonable total computation time. Finally, we put everything together and detail the full verification approach that we can use and easily adapt to verify the safety of intersection rules.

A. Sequential Path Planning for Intersections

As the basis of our approach to solving the intersection safety challenge, we propose the formalization of the Sequential Path Planning (SPP) method, which is outlined in [14], [15]. SPP is a structured approach for path planning in multi-vehicle scenarios. As suggested by the name, the key idea in SPP is to plan the paths of vehicles sequentially, prioritizing them based on a predefined order. In this method, when a higher priority vehicle $\mathcal{V}_i$ plans its path, it does so without considering subsequent vehicles. Since the path planning is sequential, when planning for a lower priority vehicle $\mathcal{V}_j$, where $i < j$, all admissible trajectories of $\mathcal{V}_i$ are already known. Consequently, $\mathcal{V}_i$ can be reserved in space and time, making it a known and deterministic obstacle for $\mathcal{V}_j$. The path planning problem for each vehicle $\mathcal{V}_j$ is then solved by computing the backward reachable set from a single-vehicle target set. After computing the backward reachable set, similar to Definition 2.1, we have a set that includes states from which $\mathcal{V}_j$ can reach its target within a specified time frame while avoiding all obstacles. To make SPP more extensible and easily adaptable, we specify an LTL formulae that we will be able to leverage to create new intersection rules.

B. LTL Specification of Sequential Path Planning

For specifying LTL formulae for SPP, we start by outlining the required specifications. First, we would like the vehicles to eventually reach their targets. Second, while they reach their targets, they should adhere to the traffic rules (speed limits, lane restrictions, etc.) in the intersection. Finally, while they reach their targets, they should also avoid colliding with other vehicles. As is done in SPP, instead of asking that the vehicles should avoid all other vehicles, we specify that it is enough that the vehicles only avoid the vehicles that are higher priority. We can write an LTL formula that collectively covers the listed specifications for an individual vehicle $\mathcal{V}_j$:

$$\varphi_j = \Diamond g_j \wedge \Box c_j \wedge \Box\neg \bigvee_{i<j} d_{j,i}. \tag{5}$$

Here, $\Diamond g_j$ corresponds to the requirement that $\mathcal{V}_j$ should eventually reach its goal, which is denoted by goal time-state set $\mathcal{G}_j \subseteq \mathbb{S}\times\mathbb{R}$. By including specific parts of $\mathcal{V}_j$'s state space in the goal set, a variety of goals can be represented, such as turning right, turning left, going straight, exiting the intersection with a specific speed, or exiting the intersection with a specific heading (as is listed in green in Fig. 2). Then, $\Box c_j$ corresponds to the requirement that $\mathcal{V}_j$ should follow traffic rules, which means its trajectories stays within a time-state constraint set $\mathcal{C}_j \subseteq \mathbb{S} \times \mathbb{R}$. Similarly to the goal set, a variety of traffic rules can be expressed through the constraint set, such as staying below the speed limit, staying in a specific lane while passing through the intersection, not allowing U-turns, and enforcing one-way street directions (as is listed in orange in Fig. 2). Finally, the last term, $\Box\neg \bigvee_{i<j} d_{j,i}$, corresponds to the requirement that $\mathcal{V}_j$ avoids collisions with higher-priority vehicles. The proposition $d_{j,i}$ corresponds to the danger time-state set $\mathcal{D}_{j,i} \subseteq \mathbb{S} \times \mathbb{R}$, which are states where $\mathcal{V}_j$ is able to collide with a higher priority vehicle $\mathcal{V}_i$. Notably, the highest priority vehicle $\mathcal{V}_1$ does not need to avoid any other vehicle. To handle this, we include a virtual vehicle $\mathcal{V}_0$ which cannot be collided with. Consequently, for any vehicle $\mathcal{V}_j$, $\mathcal{D}_{j,0} = \emptyset$ and $d_{j,0} = \mathit{false}$.

We note that the construction or computation of $\mathcal{D}_{j,i}$ is critically important to the efficiency of the intersection. When $\mathcal{D}_{j,i}$ is large, (5) will result in $\mathcal{V}_j$ driving more conservatively and, in turn, less efficiently. To maximize the efficiency of the intersection, $\mathcal{D}_{j,i}$ should closely follow the true trajectory of $\mathcal{V}_i$. However, the less conservative $\mathcal{D}_{j,i}$ is, the higher the risk that $\mathcal{V}_i$ will accidentally leave $\mathcal{D}_{j,i}$. In other words, the computation of $\mathcal{D}_{j,i}$ introduces an important trade-off between safety and efficiency. The integration and development of computational approaches to computing $\mathcal{D}_{j,i}$ is not the focus of this paper and will be addressed in future work.

C. Connecting LTL to Reachability Analysis

We will now aim to bridge the gap between the LTL specification in (5) and the subsequent reachability analyses by constructing the TLT shown in Fig. 2. We keep in mind that the collective objective is to satisfy $\varphi = \bigwedge_j \varphi_j$, yet the

actual analyses will be made sequentially for each vehicle $\mathcal{V}_j$'s objective φ_j.

1) Computing Temporal Logic Trees: As indicated by Fig. 2, the leaf nodes in the TLT represent the target proposition g_j, the state constraint proposition c_j and the collision proposition $d_{j,i}$, respectively. Through the use of the target, state constraint, and collision propositions, we are able to freely encode and adapt different desired behaviors for the intersection. Once the specification is designed, the construction of the TLT proceeds by computing the reachable tubes $\mathcal{R}(\cdot)$ and $\mathcal{RCI}(\cdot)$ underlying the "until" (U) and the "always" ($\Box$) temporal operators, respectively. The "eventually" operator is a special case of the "until" operator and, thus, is also captured by computing $\mathcal{R}(\cdot)$. Then, the presence of Boolean operators "not" ($\neg$), "or" ($\vee$), and "and" ($\wedge$) correspond to applying set complements, union, and intersection, respectively.

The application of the set operations underlying the Boolean operators is well-known and often exact. However, when an intersection is naively applied to two reachable tubes, this can result in an approximation error. This is due to the fact that the two reachable tubes may have targets or objectives that are conflicting and are not possible to simultaneously satisfy. This problem is sometimes known as the "leaking corner problem" [19]–[21]. We will refer to it as the "conflicting objectives problem" in this work. We address this conflicting objectives problem by recomputing the reachable tube of lower priority vehicles starting from the point in time where their tube intersects with the danger time-state set of higher priority vehicles. In the rest of this section, we detail the computation of the reachable tubes necessary for these SPP LTL formulae and the different computational techniques we use to reduce computational costs and avoid the conflicting objectives problem.

2) Reachability Analysis for Intersections: For intelligent intersection specifications, we construct temporal logic trees using HJ reachability analysis. The computational approach we use for HJ reachability analysis is a powerful approach that is based on finding the viscosity solution to a Hamilton-Jacobi-Isaacs Variational Inequality (HJI VI). For more details about this approach, we refer readers to [22]). HJ reachability analysis is especially beneficial for addressing the intersection safety challenge due to it's ability to easily handle the nonlinear dynamics of vehicles and non-convex road geometries. Moreover, the resultant value functions from HJ reachability analysis can be used to efficiently compute the acceleration and steering rate limits for each vehicle [19].

One of the challenges for computing reachable tubes for intersections is the handling of timing in the intersection. In particular, many problems that reachable tubes are typically computed for are time invariant problems. For example, traffic rules and other state constraints, such as one-way street directions, do not typically depend on time. However, the proposition $d_{j,i}$ is used to prevent collisions, so the corresponding $\mathcal{D}_{j,i}$ cannot be invariant since it must encode the movements of a higher priority vehicle. To address this, we adopt the double-obstacle HJI VI from [23], [24]. Specifically, consider $V_{\mathcal{G}}(z,t)$ and $V_{\mathcal{C}}(z,t)$ as two implicit surface functions representing the target and state constraints, respectively. Then the value function for $\mathcal{R}(\mathcal{G};\mathcal{C})$ becomes:

$$V(z,t) = \min_{\tau\in[t,T]} \max\{V_{\mathcal{G}}(z,\tau),\ \max_{\tau'\in[t,\tau]} V_{\mathcal{C}}(z,\tau')\}. \quad (6)$$

We use (6) to compute $\mathcal{R}(\mathcal{G};\mathcal{C}) = \{(z,t)|V(z,t) \leq 0\}$. When necessary, we compute the $\mathcal{RCI}(\cdot)$ in the same manner as is done in [19]. However, as we explain in the remaining section, there are some cases where $\mathcal{RCI}(\cdot)$ does not need to be explicitly computed, yielding a reduction of computational cost for constructing the full TLT.

D. Computational Approaches

For the remainder of this section, we describe the different computational approaches we implement for reducing the computations needed for verifying an intersection specification and for avoiding unsafe approximations when the conflicting objectives problem emerges. Then, we end by describing the full computation we perform for checking the feasibility of the intersection specification. For particular details about the implementation of these computational approaches, we refer the reader the publicly available code.

1) Simplifying the Reachability Analysis: First, to reduce the reachability analysis necessary to construct the full TLT, we find that for our particular problem, we can compute one reachable tube for each vehicle in the multi-vehicle state space. A common case for intersections is that when vehicles reach their goals in the intersection, it is the same as leaving the intersection. In these cases, we can represent the satisfaction of $\Diamond g_j \wedge \Box c_j$ with the single reachable tube $\mathcal{R}(\mathcal{G}_j;\mathcal{C}_j)$. Normally, this reachable tube only represents the satisfaction of $c_j \,\mathsf{U}\, g_j$. However, since in the case where vehicles leave the intersection when they reach their goal, to satisfy $\Diamond g_j$, we only need to keep track of trajectories that stop at $\mathcal{G}_j$ and do not need to worry about trajectories that will leave $\mathcal{G}_j$ afterwards. We find the same idea applies when including $\Box\neg\bigvee_{i<j} d_{j,i}$ and that the full specification (5) is verified by computing $\mathcal{R}(\mathcal{G}_j;\mathcal{C}_j \cap \mathcal{D}_j^C)$.

2) Avoding the Conflicting Objectives Problem: When higher priority vehicles are not present, then computing $\mathcal{R}(\mathcal{G}_j;\mathcal{C}_j)$ is sufficient to verify (5). If $\mathcal{G}_j$ and $\mathcal{C}_j$ are invariant, we can further improve performance by pre-computing the corresponding TLT subtrees for $\Diamond g_j \wedge \Box c_j$ in Fig. 2 offline. However, when introducing vehicles to the intersection, we need to ensure that the conflicting objectives problem is avoided to fully ensure the safety of all vehicles. To do this, consider the time frame $T_{\mathcal{D}_j} \subseteq \mathbb{R}$ during which a vehicle $\mathcal{V}_j$ interact with higher priority vehicles and the conflicting objectives problem emerges. The time frame is a closed interval $T_{\mathcal{D}_j} = [t_a, t_b]$. For any $t > t_b$ it is sufficient to verify $\Diamond g_j \wedge \Box c_j$ since $\mathcal{D}_j$ will be empty and, consequently, $\Box\neg d_{j,i}$ will be true. Assuming the target and state constraints are invariant, then this can be done offline. On the other hand, for any $t \leq t_b$ we need to recompute the analysis with the added collision constraint that ensures a safe interaction

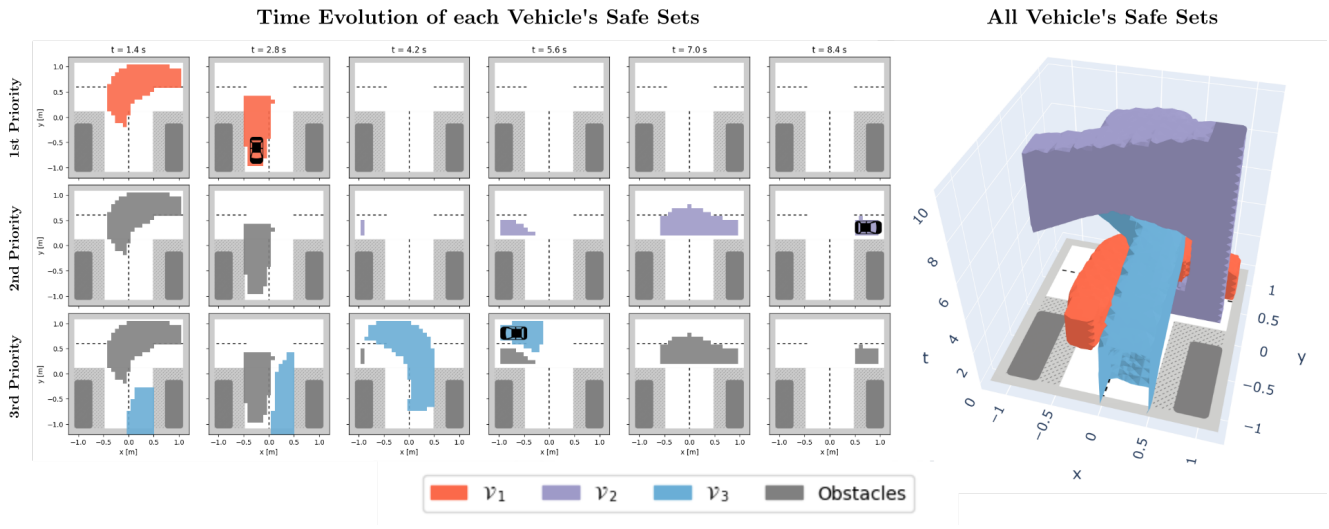

Fig. 3. Shown are the safe sets for all vehicles over time (left part) and the sliced time snapshots of the safe sets for each vehicle, $\mathcal{V}_1$ (red), $\mathcal{V}_2$ (blue) and $\mathcal{V}_3$ (purple), with priority in the given order (right part). Since the safe sets are computed starting from the goal state, we mark the goal state of each vehicle with an icon. The top row of the right part shows the analysis done for $\mathcal{V}_1$ w.r.t its individual objective φ_1. Similarly, the second and third rows show the analyses done for $\mathcal{V}_2$ and $\mathcal{V}_3$, respectively, where the gray regions are the reachable sets of higher priority vehicles seen as obstacles.

with higher priority vehicles. By doing this, we reduce the total amount of reachability analysis to performed online and avoid the conflicting objectives problem.

3) Full Computation: Finally, our approach to computing the TLT in Figure 2 is the following. We start by computing $\mathcal{R}(\mathcal{G}_j; \mathcal{C}_j)$ for vehicle $\mathcal{V}_j$ offline and store it in memory. We call this step the "Offline Pass". Then, we perform a second reachability analysis online, which we call the "Online Pass". Specifically, the online pass updates the solution of $\mathcal{R}(\mathcal{G}_j; \mathcal{C}_j)$ over $t \leq t_b$ with

$$\mathcal{R}(\mathcal{G}_j \cup \{(z, t_b) \mid z \in \Omega_{\mathcal{R}(\mathcal{G}_j, \mathcal{C}_j)}(t_b)\};\ \mathcal{C}_j \cap \mathcal{D}_j^C). \quad (7)$$

Incorporating these two passes enables us to efficiently check the satisfaction of the intersection specification.

In summary, by using this approach, we develop a verification method where we can freely design and adapt the requirements on the behavior of the vehicles passing through the intersection using the expressiveness of temporal logic. Then, by constructing temporal logic trees using HJ reachability analysis, we can handle general nonlinear dynamics and complex constraints in the state space of the vehicle model. In the next section, we evaluate the practicalities of applying this method to a T-intersection example.

IV. Numerical Results

In this section the T-intersection scenario, as described in Section II-D, is presented in simulation. The simulation, shown in Fig. 3, includes three vehicles that cross the intersection in a way that risk collision if there is no coordination between them. Each vehicle is modelled by (1) with working space and control constraints: $\mathbb{S}_i = \{z_i \in \mathbb{R}^5 \mid -1.2 \leq x_i \leq 1.2, -1.2 \leq y_i \leq 1.2, -\pi \leq \theta_i \leq \pi, -\pi/5 \leq \delta_i \leq \pi/5, 0 \leq v_i \leq 1\}$, $\mathbb{U}_i = \{u_i \in \mathbb{R}^2 \mid -\pi \leq s_i \leq \pi, -0.5 \leq a_i \leq 0.5\}$. The roads enforce constraints on the vehicles' heading, except in the middle of the intersection. Furthermore, a lower-bound speed limit of 0.4 m/s is set to prevent the vehicles from stopping and blocking the way for following vehicles. Finally, for each road we define entry and exit targets. For example, $\mathcal{V}_2$ will enter at $\mathbb{G}_2^{enter} = \{z_2 \in \mathbb{S}_2 \mid 0 \leq x_2 \leq 0.5, -1.2 \leq y_2 - 0.7\}$. The reachability analysis is performed using the Python package `hj_reachability`[2] which solves the HJI VI on a $31 \times 31 \times 31 \times 7 \times 11$ grid of the discretized single-vehicle state space $\mathbb{S}_i$. `hj_reachability` can compute this on the GPU. In Table I we show how long these operations take on an NVIDIA GeForce RTX 2080 Ti.

In Figure 3, we show the time evolution of the computed safe sets of each vehicle. The analysis' time horizon starts at $t = 0$, which is the current time, and ends at $t = 10$. The top row shows the evolution of the highest priority vehicle. The plotted safe set corresponds to all of the locations the highest priority vehicle can be in to satisfy the intersection specification. Then, after the safe set of the highest priority vehicle is computed, the safe set is passed to the computation of the next vehicle to be used as a danger set (grey set). We repeat this for each lower priority vehicle. In other words, the red and purple sets are the danger sets ($\mathcal{D}_{3,1}$ and $\mathcal{D}_{3,2}$) for the 2nd and 3rd priority vehicles, respectively. Interestingly, we see the reachable set Θ_3 as computed in the online pass for $\mathcal{V}_3$ in blue. Here, we clearly see how the reachability analysis ensures that $\mathcal{V}_3$ avoid the higher priority vehicles. The online pass is computed over the time frame $T_{\mathcal{D}_3} = [0, 6]$ during which it removes states that would otherwise lead to a collision with either $\mathcal{V}_1$ or $\mathcal{V}_2$.

These results provide preliminary indication that the presented method can be used in practical settings. Although the computation times reported in Table I are not fast enough to be used in cases where the verification should occur while vehicles are in the intersection, the computations are fast

[2]https://github.com/StanfordASL/hj_reachability

Vehicle	Offline Pass [s]	Online Pass [s]
$\mathcal{V}_1$	16.11	0.0
$\mathcal{V}_2$	13.68	6.71
$\mathcal{V}_3$	13.17	8.25

TABLE I

COMPUTATIONAL TIME OF THE REACHABILITY ANALYSES FOR EACH VEHICLE ON NVIDIA GEFORCE RTX 2080 TI.

enough if the vehicle has not arrived to the intersection yet. Moreover, in this work, we do not explore a variety of computational techniques that can further optimize the computation time for verifying the intersection safety. For example, in the case where a vehicle needs to be rescheduled, the tubes illustrated on the right-side of Fig. 3 could simply be moved up or down the time-axis at almost no computational cost. Furthermore, due to the richness of the value function underlying the safe sets, the valid acceleration and steering rates the vehicles should implement can also be computed with a very low computational complexity [19].

V. CONCLUSION

In this work, we propose a framework based on LTL and Hamilton-Jacobi reachability analysis for ensuring the safety at intelligent intersections. By formalizing SPP into LTL formulae, we leverage temporal logic trees to break down the intersection safety problem into a series of Hamilton-Jacobi reachability analyses. Due to this approach, the safety framework is able to handle changes in the intersection specification, while maintaining safety guarantees. We illustrate the framework's utility on a simulated T-intersection example, where we show we are able to verify that the vehicles can pass through the intersection safely. While we include several optimizations in our implementation to reduce the total computational times in the T-intersection example, an important future work will be to further reduce the computation time by employing techniques for directly addressing the conflicting objectives problem, such as the technique presented in [21]. Moreover, we are building on the theoretical foundation of the presented framework to design and implement an intelligent intersection management system that integrates scheduling and V2I communication, so we can use the testbed presented in [25] to evaluate the framework's performance with real hardware and communication networks in the loop.

REFERENCES

[1] O. Grembek, A. Kurzhanskiy, A. Medury, P. Varaiya, and M. Yu, "Introducing an Intelligent Intersection," *UC ITS Reports*, vol. 13, 2018.

[2] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Autonomous Agents and Multiagent Systems, International Joint Conference on*, vol. 3. IEEE Computer Society, 2004, pp. 530–537.

[3] E. Namazi, J. Li, and C. Lu, "Intelligent Intersection Management Systems Considering Autonomous Vehicles: A Systematic Literature Review," *IEEE Access*, vol. 7, pp. 91 946–91 965, 2019.

[4] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 145–154.

[5] X. Chen and J. Mårtensson, "Heterogeneous Traffic Intersection Coordination Based on Distributed Model Predictive Control with Invariant Safety Guarantee," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3617–3624.

[6] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable Systemwide Safety in Intelligent Intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, 2011.

[7] J. Thunberg, G. Sidorenko, K. Sjöberg, and A. Vinel, "Efficiently Bounding the Probabilities of Vehicle Collision at Intelligent Intersections," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 2, pp. 47–59, 2021.

[8] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[9] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of Intersection Traffic Rules in Temporal Logic," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1135–1144.

[10] M. Saraoglu, J. Pintscher, and K. Janschek, "Designing a Safe Intersection Management Algorithm using Formal Methods," *IFAC-PapersOnLine*, vol. 55, no. 14, pp. 22–27, 2022.

[11] J. Haydon, M. Bondu, C. Eberhart, J. Dubut, and I. Hasuo, "Formal Verification of Intersection Safety for Automated Driving," *arXiv preprint arXiv:2308.06785*, 2023.

[12] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," aug 2017. [Online]. Available: http://arxiv.org/abs/1708.06374

[13] Y. Gao, A. Abate, F. J. Jiang, M. Giacobbe, L. Xie, and K. H. Johansson, "Temporal Logic Trees for Model Checking and Control Synthesis of Uncertain Discrete-Time Systems," *IEEE Transactions on Automatic Control*, vol. 67, no. 10, pp. 5071–5086, oct 2022.

[14] M. Chen, S. Bansal, K. Tanabe, and C. J. Tomlin, "Provably Safe and Robust Drone Routing via Sequential Path Planning: A Case Study in San Francisco and the Bay Area," no. arXiv:1705.04585, may 2017. [Online]. Available: http://arxiv.org/abs/1705.04585

[15] S. Bansal, M. Chen, K. Tanabe, and C. J. Tomlin, "Provably Safe and Scalable Multivehicle Trajectory Planning," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2473–2489, nov 2021.

[16] T. Wongpiromsarn, *Formal methods for design and verification of embedded control systems: application to an autonomous vehicle*. California Institute of Technology, 2010.

[17] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–9.

[18] P. Yu, Y. Gao, F. J. Jiang, K. H. Johansson, and D. V. Dimarogonas, "Online control synthesis for uncertain systems under signal temporal logic specifications," *The International Journal of Robotics Research*, 2023.

[19] F. J. Jiang, K. M. Arfvidsson, C. He, M. Chen, and K. H. Johansson, "Guaranteed completion of complex tasks via temporal logic trees and hamilton-jacobi reachability," *arXiv preprint arXiv:2404.08334*, 2024.

[20] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[21] C. He, Z. Gong, M. Chen, and S. Herbert, "Efficient and Guaranteed Hamilton-Jacobi Reachability via Self-Contained Subsystem Decomposition and Admissible Control Sets," *IEEE Control Systems Letters*, 2023.

[22] M. Chen and C. J. Tomlin, "Hamilton–Jacobi Reachability: Some Recent Theoretical Advances and Applications in Unmanned Airspace Management," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 333–358, may 2018.

[23] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. Seattle Washington: ACM, apr 2015, pp. 11–20.

[24] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin, "Safe Sequential Path Planning of Multi-Vehicle Systems via Double-Obstacle Hamilton-Jacobi-Isaacs Variational Inequality," no. arXiv:1412.7223, mar 2016. [Online]. Available: http://arxiv.org/abs/1412.7223

[25] K. M. Arfvidsson, K. Fragkedaki, F. J. Jiang, V. Narri, H.-C. Lindh, K. H. Johansson, and J. Mårtensson, "Small-Scale Testbed for Evaluating C-V2X Applications on 5G Cellular Networks," in *2024 IEEE Intelligent Vehicles Symposium (IV)*.