2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Formal Verification of Linear Temporal Logic Specifications Using Hybrid Zonotope-Based Reachability Analysis

Loizos Hadjiloizou[1], Frank J. Jiang[1], Amr Alanwar[2], Karl H. Johansson[1]

***Abstract*— In this paper, we introduce a hybrid zonotope-based approach for formally verifying the behavior of autonomous systems operating under Linear Temporal Logic (LTL) specifications. In particular, we formally verify the LTL formula by constructing temporal logic trees (TLT)s via backward reachability analysis (BRA). In previous works, TLTs are predominantly constructed with either highly general and computationally intensive level set-based BRA or simplistic and computationally efficient polytope-based BRA. In this work, we instead propose the construction of TLTs using hybrid zonotope-based BRA. By using hybrid zonotopes, we show that we are able to formally verify LTL specifications in a computationally efficient manner while still being able to represent complex geometries that are often present when deploying autonomous systems, such as non-convex, disjoint sets. Moreover, we evaluate our approach on a parking example, providing preliminary indications of how hybrid zonotopes facilitate computationally efficient formal verification of LTL specifications in environments that naturally lead to non-convex, disjoint geometries.**

I. INTRODUCTION

Advancements in fields like computer science, hybrid systems, and cyber-physical systems have ushered in a new era of complexity. As we extend the limits of what's possible, ensuring the safety of ever more complex systems becomes crucial. This endeavor requires adept methods for describing and analyzing their complexity. A promising approach is to leverage temporal logic formalisms [1] to capture system objectives and then use reachability analysis [2] to study its behaviour.

A significant body of literature employing this approach, focuses on automata-based methods [1], [3]. While powerful, automata tend to suffer from computational and expressive limitations when applied to continuous state-space systems with time-varying specifications or environments [4]. In this work, we further explore an alternative to automata first proposed in [4], based on a tree structure called Temporal Logic Tree to encode information about the satisfaction of an LTL specification. Unlike automata, they are abstraction-free for continuous state-space systems, support the full LTL language, and are more modular, enabling online adaption to time-varying specifications or environments.

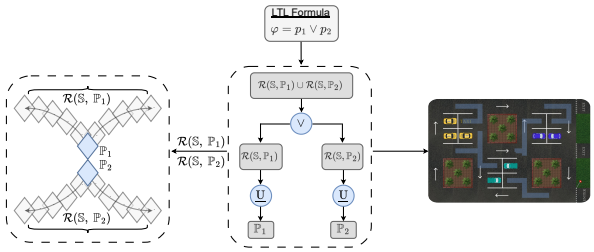


Fig. 1: We illustrate an overview of our approach.

The computational complexity of constructing a TLT depends largely on the chosen BRA technique. In [5], Hamilton-Jacobi reachability analysis is used to build a TLT, ensuring safety in vehicle parking tasks. This method facilitates the construction of a TLT for systems with nonlinear dynamics, and through a level-set representation, for environments with intricate geometries. While powerful and general, Hamilton-Jacobi reachability analysis' computational complexity grows exponentially with the system's state space [6]. More computationally efficient tools for reachability analysis do exist, such as polytope, zonotope ($\mathcal{Z}$), constrained zonotope ($\mathcal{CZ}$), sparse polynomial zonotope ($\mathcal{SPZ}$), and constrained polynomial zonotope ($\mathcal{CPZ}$)-based methods [7], [8], [9], [10], [11]. While these approaches can represent geometries of varying complexity, to the extent of our knowledge, they can not efficiently handle disjoint sets, which is often crucial for autonomous systems. A recent work proposed a new set representation called hybrid zonotope ($\mathcal{HZ}$) that, among other benefits, can efficiently handle disjoint sets [12]. By leveraging $\mathcal{HZ}$-based BRA in this work, we aim to efficiently verify LTL specifications that include disjunction operators in, for example, a parking lot environment where the reachable sets naturally become disjoint [5].

The main contribution of this work is a verification approach that uses $\mathcal{HZ}$-based BRA to construct TLTs in a computationally efficient-manner. This allows for the efficient verification of the feasibility of any LTL formula in environments with relatively challenging geometries. More specifically, the contributions are outlined as follows: (1) detailing the construction of TLTs using $\mathcal{HZ}$-based BRA, (2) implementing the proposed method, which is publicly available at [1], and (3) evaluating the implemented approach on a practical example with an LTL specification containing

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, and the Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant.

[1]L. Hadjiloizou, F. J. Jiang, and K. H. Johansson are with the Division of Decision and Control Systems, EECS, KTH Royal Institute of Technology, Malvinas väg 10, 10044 Stockholm, Sweden, {loizosh, frankji, kallej}@kth.se. They are also affiliated with Digital Futures.

[2]A. Alanwar is with the School of Computation, Information and Technology, Technical University of Munich, Bildungscampus 2, 74076 Heilbronn, Germany, alanwar@tum.de

[1]https://github.com/loizoshad/zonopy

disjunction operators and disjoint reachable sets.

The remainder of this paper is structured as follows: We introduce preliminary material on reachability analysis, LTL, and TLTs in Section II. Section III formally introduces the problem solved in this work as well as provides a motivation for choosing $\mathcal{HZ}$s. Section IV lays out our approach for constructing a TLT using $\mathcal{HZ}$s. In Section V, we formulate a parking case study. Section VI analyzes the results from our simulations and finally, Section VII, concludes our work and discusses some future work that can be done in this area.

II. PRELIMINARIES

A. Notation

We denote the set of real numbers as $\mathbb{R}$, vectors as bold lowercase letters (e.g., $\boldsymbol{x} \in \mathbb{R}^n$), the identity matrix by I, while matrices filled with elements 1 and 0 are represented by $\mathbf{1}, \mathbf{0}$, respectively. We extract a submatrix from G by selecting its first n rows as $G[1:n,:]$ and express the horizontal concatenation of matrices A and B as $[A\ B]$. A diagonal matrix is denoted by $diag(\boldsymbol{v})$, with $\boldsymbol{v}$ being the vector with the diagonal elements. The n-dimensional and constrained n-dimensional unit hypercubes are denoted by $\mathcal{B}_\infty^n = \{\boldsymbol{\xi} \in \mathbb{R}^n | \ ||\boldsymbol{\xi}||_\infty \leq 1\}$ and $\mathcal{B}_\infty^n(A, \boldsymbol{b}) = \{\boldsymbol{\xi} \in \mathbb{R}^n | \ ||\boldsymbol{\xi}||_\infty \leq 1, A\boldsymbol{\xi} = \boldsymbol{b}\}$, respectively, while the power set of an n-dimensional vector of binary variables as $\{-1, 1\}^n$.

B. Plant Model

Consider the discrete-time system

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{n_x}$ represents the state vector, $\boldsymbol{u}_t \in \mathbb{R}^{n_u}$ the control input, and the matrices $A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}$, are the state transition and control input matrices, respectively. For each time instant t, $u_t \in \mathbb{U} \subset \mathbb{R}^{n_u}$, where $\mathbb{U}$ is the compact control input constraint. We assume the system is well-defined such that x_{t+1} is uniquely determined by x_t and u_t. Let $\mu = u_0, u_1, \ldots$ be a control policy and $\mathcal{M}$ be the set of all control policies. Then, given initial state x_0, we denote a trajectory of system (1) as $\zeta(\cdot; x_0, \mu)$ where $\zeta(t; x_0, \mu)$ is the state of system (1) at time t when starting from x_0 and implementing control policy μ. For simplicity, we will sometimes refer to trajectories with $\zeta(\cdot)$.

C. Reachability Analysis

Reachability analysis is a common approach to formally provide guarantees about the behavior of a system. In this subsection, we provide the relevant reachability definition that will later be used for the construction of TLTs.

Definition 2.1: Consider the plant (1), then the infinite horizon backward reachable set (BRS) from the target set $\mathcal{T} \subseteq \mathbb{R}^{n_x}$ in state space $\mathbb{S} \subseteq \mathbb{R}^{n_x}$ is given as

$$\mathcal{R}(\mathbb{S}, \mathcal{T}) = \{x \in \mathbb{S} | \exists \mu \in \mathcal{M}, \exists t > 0, \zeta(t; x, \mu) \in \mathcal{T}\}. \tag{2}$$

Intuitively, the infinite horizon BRS answers the question of where the system can begin and eventually reach the target set $\mathcal{T}$. This definition is in accordance with the controlled reachable set definition in [4]. As it will become apparent later, it allows us to construct the controlled TLT, which for the sake of brevity, we generally refer to as TLT in this work.

In practice, to compute the infinite horizon BRS, one needs to compute the union of all N-step BRSs for $N = [0, \infty), N \in \mathbb{Z}$, where the N-step BRS from the target set $\mathcal{T} \subseteq \mathbb{R}^{n_x}$ in $\mathbb{S} \subseteq \mathbb{R}^{n_x}$ is the set of all states for which system (1) can reach the target set in exactly N time steps. The N-step BRS is equivalent to computing the predecessor set N times, (e.g., $\mathcal{R}(\mathbb{S}, \mathcal{T}, 3) = \mathcal{P}(\mathbb{S}, \mathcal{P}(\mathbb{S}, \mathcal{P}(\mathbb{S}, \mathcal{T})))$) which in turn is defined as

$$\mathcal{P}(\mathbb{S}, \mathcal{T}) = \{x \in \mathbb{S} | \exists u \in \mathbb{U}, Ax + Bu \in \mathcal{T}\}. \tag{3}$$

D. Linear Temporal Logic

In the context of this work, LTL plays a pivotal role in modeling complex temporal tasks to study the temporal and time-invariance properties of autonomous systems. At its core, an LTL formula consists of three components: a finite set of atomic propositions (P), a set of temporal ($until$: $\underline{U}$), and logical ($negation := \neg$, $and := \wedge$) operators. In this work, we consider the following LTL syntax:

$$\varphi ::= \ true \mid p \in \mathsf{P} \mid \neg\phi \ \mid \phi_1 \wedge \phi_2 \ \mid \phi_1 \underline{U} \phi_2 \mid \bigcirc \phi. \tag{4}$$

Definition 2.2: For an LTL formula φ, a labeling function $l(\cdot)$, a trajectory $\zeta(\cdot)$, and a time instant $t \geq 0$, the satisfaction relation $(\zeta(\cdot), t) \vDash \varphi$ is defined as

$$\begin{aligned}
&(\zeta(\cdot), t) \vDash p \in \mathsf{P} \Leftrightarrow \zeta(t) \in l(p),\\
&(\zeta(\cdot), t) \vDash \neg\varphi \Leftrightarrow (\zeta(\cdot), t) \nvDash \varphi,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow (\zeta(\cdot), t) \vDash \varphi_1 \wedge (\zeta(\cdot), t) \vDash \varphi_2,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow (\zeta(\cdot), t) \vDash \varphi_1 \vee (\zeta(\cdot), t) \vDash \varphi_2,\\
&(\zeta(\cdot), t) \vDash \Diamond\varphi \Leftrightarrow \exists t_1 \in [t, \infty) : (\zeta(\cdot), t_1) \vDash \varphi,\\
&(\zeta(\cdot), t) \vDash \Box\varphi \Leftrightarrow \forall t_1 \in [t, \infty) : (\zeta(\cdot), t_1) \vDash \varphi,\\
&(\zeta(\cdot), t) \vDash \varphi_1 \underline{U} \varphi_2 \Leftrightarrow \exists\ t_1 \in [t, \infty) : (\zeta(\cdot), t_1) \vDash \varphi_2,\\
&\qquad\qquad \forall t_2 \in [t, t_1), (\zeta(\cdot), t_2) \vDash \varphi_1,\\
&(\zeta(\cdot), t) \vDash \bigcirc\varphi \Leftrightarrow (\zeta(\cdot), t+1) \vDash \varphi.
\end{aligned} \tag{5}$$

E. Temporal Logic Tree

A TLT is a hierarchical structure that describes the satisfaction relationship between the system's state space and LTL specification. This structure serves as an alternative approach for model checking and control synthesis, which provides high modularity. Another significant characteristic of the TLT is that it enables the synthesis of a controller with multiple control policies, which is useful for applications with mixed levels of autonomy, such as autonomous vehicles.

Definition 2.3: A TLT is a tree for which

- each node is either a set node within $\mathbb{R}^{n_x}$ or an operator node from the LTL syntax defined in (4);
- the root and leaf nodes are set nodes;
- a set node that is not a leaf node has as a unique child an operator node;
- all operator nodes have set nodes are their children.

Example 2.1: Given the dynamic model of an autonomous vehicle described by (1), study the eligibility of all states to allow the vehicle to eventually reach one of the available

parking spots ($\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_5$). Given this description, we can formulate the following LTL to capture this goal.

$$\varphi = (\Diamond p_1) \vee (\Diamond p_2) \vee (\Diamond p_3) \vee (\Diamond p_4) \vee (\Diamond p_5), \quad p_i = \{\boldsymbol{x} \in \mathbb{S} | \boldsymbol{x} \in \mathbb{P}_i\}, \quad i = \{1, 2, ..., 5\}. \tag{6}$$

- **Step 1**: Obtain weak-until positive normal form (PNF) [1]. The PNF for LTL permits negation only on the level of atomic propositions. To ensure the full expressiveness of LTL, in addition to the LTL fragment in (4) it includes the Boolean *Or* $\vee$ operator as well us the *Weak-Until* $\underline{W}$ operator, where $\varphi_1 \underline{W} \varphi_2 \equiv \varphi_1 \underline{U} \varphi_2 \vee \Box \varphi_1$.

$$\varphi_N = (true\, \underline{U}\, p_1) \vee \cdots \vee (true\, \underline{U}\, p_5). \tag{7}$$

- **Step 2**: Compute the TLT for each atomic proposition. The TLT for any atomic proposition is simply a set node consisting of all states that satisfy it. In this example, these nodes are the set nodes, each containing the equivalent parking area $\mathbb{P}_i$ for every parking spot i. These are the leaf nodes of the TLT in Figure 2.
- **Step 3**: Inductively construct the TLT for each component of the formula. Each branch in Figure 2 represents the TLT for each component $(true\, \underline{U}\, p_i)$. Every branch consists of two set nodes and one operator node. The node at the bottom is the target i the vehicle needs to reach, while the top set node is the set of all states that could eventually take the vehicle to target i, hence the backward reachable set of the target $\mathbb{P}_i$. These two nodes are connected by the *Until* temporal operator. Intuitively, each branch asks that the vehicle is within the BRS of each target until it reaches the target itself. After combining all the branches' backward reachable sets through set unions, the root node consists of all the states for which we can guarantee satisfaction of the LTL specification.

This example hints that the effectiveness of TLTs in formally verifying LTL specifications relies on the set representation's support for different operators.

III. Problem Statement

Constructing a TLT for any LTL problem involves computations such as BRSs, intersections, and unions. In many cases, this results in non-convex, disjoint sets. While level-set Hamilton Jacobi reachability can handle such complexities,

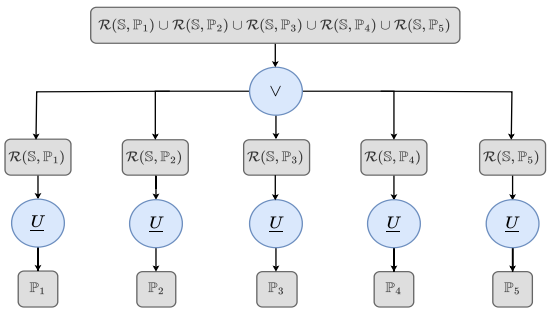


Fig. 2: TLT to check if the vehicle can eventually park.

	Polytope	$\mathcal{Z}$	$\mathcal{CZ}$	$\mathcal{SPZ}$	$\mathcal{CPZ}$	$\mathcal{HZ}$	Level Set
Linear Transformation	✓	✓	✓	✓	✓	✓	✓
Minkowski Sum	✓	✓	✓	✓	✓	✓	✓
Intersection	✓	✗	✓	✗	✓	✓	✓
Union	✗	✗	✗	✗	✓	✓	✓
Non-Convex	NO	NO	NO	YES	YES	YES	YES
Disjoint	NO	NO	NO	NO	NO	YES	YES

TABLE I: Properties of set representation tools[2]

it comes with increased computational expense. Conversely, more computationally efficient methods can construct TLTs in relatively simple scenarios but to the best of our knowledge, can not efficiently handle the full LTL language requirements. Table I provides an overview of common set representations for reachability analysis and their suitability for addressing these requirements, among which, only level sets and $\mathcal{HZ}$s can effectively handle the full LTL language. Due to the computational complexity of level-set solutions, we explore the use of $\mathcal{HZ}$s in this work and define the following problem statement.

Problem 3.1: For an autonomous system with the dynamics (1) and an LTL specification φ, formally study the satisfiability of φ by (1) in a computationally efficient manner for environments that naturally form non-convex, disjoint sets.

IV. Constructing TLTs using Hybrid Zonotopes

In this section, we introduce and formulate our approach for constructing TLTs using hybrid zonotopes.

A. Hybrid Zonotopes

To understand the $\mathcal{HZ}$, let's start by defining its predecessor, the zonotope which is a centrally symmetric convex polytope, defined as the affine image of a unit hypercube.

Definition 4.1: A set $\mathcal{Z} \subset \mathbb{R}^{n_x}$ is a zonotope if there exist a matrix $G \in \mathbb{R}^{n_x \times n_g}$ and a vector $\boldsymbol{c} \in \mathbb{R}^{n_x}$, where n_g the number of generators, such that $\mathcal{Z} = \{\boldsymbol{c} + G\boldsymbol{\xi} \mid ||\boldsymbol{\xi}||_\infty \leq 1\}$.

The constrained zonotope is an immediate extension of the zonotope, which allows linear equality constraints to be imposed on its coefficients $\boldsymbol{\xi}$. This results in a non-centrally symmetric convex polytope.

Definition 4.2: A set $\mathcal{CZ} \subset \mathbb{R}^{n_x}$ is a constrained zonotope if there exist matrices $G \in \mathbb{R}^{n_x \times n_g}, A \in \mathbb{R}^{n_c \times n_g}$ and vectors $\boldsymbol{c} \in \mathbb{R}^{n_x}, \boldsymbol{b} \in \mathbb{R}^{n_c}$, where n_c is the number of linear equality constraints, such that $\mathcal{CZ} = \{\boldsymbol{c} + G\boldsymbol{\xi} \mid ||\xi||_\infty \leq 1, A\boldsymbol{\xi} = \boldsymbol{b}\}$.

The constrained zonotope is represented in its constrained generator form and, in short, denoted as $\mathcal{CZ} = \{\boldsymbol{c}, G, A, \boldsymbol{b}\}$.

The hybrid zonotope extends the notion of the constrained zonotope by restricting the coefficients of some generators to lie only on the edges of the unit hypercube. These generators are called the binary generators of the $\mathcal{HZ}$. This can create gaps in the set due to the absence of entries from the

[2]✓: Set representation closed under the operation, ✗: Set representation not closed under the operation

TABLE II: Corresponding operation for the LTL syntax

LTL	Set Operation for TLT Construction
$p \in \mathsf{P}$	$\mathcal{HZ}_{p\in\mathsf{P}} = \{s \in \mathbb{P}\}$
$true$	$\mathcal{HZ} = \{s \in \mathbb{S}\}$
$\neg\phi$	$\mathcal{HZ} = compl(\mathcal{CZ})$, [13, eq. (16)]
$\phi_1 \wedge \phi_2$	$\mathcal{HZ}_{\phi_1\wedge\phi_2} = inters(\mathcal{HZ}_{\phi_1}, \mathcal{HZ}_{\phi_2})$, [12, eq. (8)]
$\phi_1 \vee \phi_2$	$\mathcal{HZ}_{\phi_1\vee\phi_2} = union(\mathcal{HZ}_{\phi_1}, \mathcal{HZ}_{\phi_2})$, [13, eq. (4)]
$\phi_1 \underline{U} \phi_2$	$\mathcal{HZ}_{\phi_1\underline{U}\phi_2} = \mathcal{R}(\mathcal{HZ}_{\phi_1}, \mathcal{HZ}_{\phi_2})$
$\phi_1 \underline{W} \phi_2$	$\mathcal{HZ}_{\phi_1\underline{W}\phi_2} = union(\mathcal{RCI}(\mathcal{HZ}_{\phi_1}), \mathcal{HZ}_{\phi_2})$
$\Diamond\phi = true\underline{U}\phi$	$\mathcal{HZ}_{true\underline{U}\phi} = \mathcal{R}(\mathbb{S}, \mathcal{HZ}_{\phi})$
$\Box\phi = \phi\underline{W}false$	$\mathcal{HZ}_{\phi\underline{W}false} = \mathcal{RCI}(\mathcal{HZ}_{\phi})$
$\bigcirc\phi$	$\mathcal{HZ}_{\bigcirc\phi} = \mathcal{P}(\mathbb{S}, \mathcal{HZ}_{\phi})$

remaining coefficients in the range $(-1, 1)$. Intuitively, a $\mathcal{HZ}$ is equivalent to the union of 2^{n_b} constrained zonotopes [12], where n_b is the number of binary generators of the $\mathcal{HZ}$.

Definition 4.3: A set $\mathcal{HZ} \subset \mathbb{R}^{n_x}$ is a hybrid zonotope if there exist matrices $G^c \in \mathbb{R}^{n_x\times n_g}, G^b \in \mathbb{R}^{n_x\times n_b}, A^c \in \mathbb{R}^{n_c\times n_g}, A^b \in \mathbb{R}^{n_c\times n_b}$ and vectors $\boldsymbol{c} \in \mathbb{R}^{n_x}, \boldsymbol{b} \in \mathbb{R}^{n_c}$ such that

$$\mathcal{HZ} = \left\{ \begin{bmatrix} G^c & G^b \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^c \\ \boldsymbol{\xi}^b \end{bmatrix} + \boldsymbol{c} \,\middle|\, \begin{bmatrix} \boldsymbol{\xi}^c \\ \boldsymbol{\xi}^b \end{bmatrix} \in \mathcal{B}_\infty^{n_g} \times \{-1, 1\}^{n_b}, \begin{bmatrix} A^c & A^b \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^c \\ \boldsymbol{\xi}^b \end{bmatrix} = \boldsymbol{b} \right\}. \tag{8}$$

The hybrid zonotope is represented in its hybrid constrained generator form and is, in short, denoted as $\mathcal{HZ} = \{\boldsymbol{c}, G^c, G^b, A^c, A^b, \boldsymbol{b}\}$.

B. TLT Construction

Now, it is time to show how one can use the $\mathcal{HZ}$ to verify whether an LTL specification can be satisfied. We start by defining an LTL formula whose feasibility we want to verify. Then we obtain its PNF and define an $\mathcal{HZ}$ for each of its atomic propositions such that it consists all the states that satisfy that particular atomic proposition. Then, we inductively construct each sub-TLT corresponding to a temporal or logical operation. Starting with the logical *Or* and *And* operations, the construction of their TLTs requires the computation of the union and the intersection of two $\mathcal{HZ}s$ respectively. Meanwhile, the construction of a TLT for the *Until* and *Weak-Until* operators requires the computation of unions, backward reachable sets, as well as robust controlled invariant (RCI) sets for the latter operation. As shown in [2], the computation of an RCI is a series of predecessor and intersection set operations. Following Table II, the TLT for any LTL formula can be constructed. In Table II, $\mathcal{HZ}_\phi$ is the computed set for satisfying sub-formula ϕ and $\mathcal{RCI}(\phi) = \cap_{i=0}^{\infty}\mathcal{R}(\mathbb{S}, \mathcal{HZ}_\phi)$.

To compute the operations for Table II, we need to compute the BRS for system (1) using $\mathcal{HZ}s$. Since we are dealing with discrete-time linear dynamics, we can use the following formula the formula developed in [14] for computing the predecessor set (3):

The predecessor set from a target set $\mathcal{T} = \{\boldsymbol{c}_t, G_t^c, G_t^b, A_t^c, A_t^b, \boldsymbol{b}_t\} \subseteq \mathbb{R}^{n_x}$ from the augmented state space $\hat{\mathbb{S}} = \{\boldsymbol{c}_s, G_s^c, G_s^b, A_s^c, A_s^b, \boldsymbol{b}_s\} \subseteq \mathbb{R}^{n_x+n_u}$ is defined as $\mathcal{B} = \{\boldsymbol{c}_b, G_b^c, G_b^b, A_b^c, A_b^b, \boldsymbol{b}_b\}$, where

$$G_b^c = \begin{bmatrix} G_s^c[1:n_x,:] & \mathbf{0} \end{bmatrix}, \quad A_b^c = \begin{bmatrix} A_s^c & \mathbf{0} \\ \mathbf{0} & A_t^c \\ [A\ B]G_s^c & -G_t^c \end{bmatrix},$$
$$G_b^b = \begin{bmatrix} G_s^b[1:n_x,:] & \mathbf{0} \end{bmatrix}, \quad A_b^b = \begin{bmatrix} A_s^b & \mathbf{0} \\ \mathbf{0} & A_t^b \\ [A\ B]G_s^b & -G_t^b \end{bmatrix},$$
$$\boldsymbol{c}_b = \boldsymbol{c}_s[1:n_x,:], \qquad \boldsymbol{b}_b = \begin{bmatrix} \boldsymbol{b}_s \\ \boldsymbol{b}_t \\ \boldsymbol{c}_t - [A\ B]\boldsymbol{c}_s \end{bmatrix}. \tag{9}$$

Algorithm 1 Constructing TLT using $\mathcal{HZ}$

Input: φ_N ▷ LTL specification
$\hat{\mathbb{S}} = \{\boldsymbol{c}_s, G_s^c, G_s^b, A_s^c, A_s^b, \boldsymbol{b}_s\}$, ▷ Augm. State Space
$\mathbb{P}_1 = \{\boldsymbol{c}_t, G_t^c, G_t^b, A_t^c, A_t^b, \boldsymbol{b}_t\}$, ▷ Target Space
$D = [A\ B]$, ▷ System Dynamics
Output: $\mathcal{R}(\hat{\mathbb{S}}, \mathbb{P}_1)$ ▷ Root Set Node
Process:
$\mathcal{R}(\hat{\mathbb{S}}, \mathbb{P}_1) \leftarrow \mathbb{P}_1$ ▷ Init BRS
$Pre \leftarrow \mathbb{P}_1$ ▷ Init Predecessor Set
while True **do**
  $Pre \leftarrow \mathcal{P}(\hat{\mathbb{S}}, Pre)$ ▷ Predecessor Set
  if $Pre \subseteq \mathcal{R}(\hat{\mathbb{S}}, \mathbb{P}_1)$ **then**
    break
  end if
  $\mathcal{R}(\hat{\mathbb{S}}, \mathbb{P}_1) \leftarrow \mathcal{R}(\hat{\mathbb{S}}, \mathbb{P}_1) \cup Pre$
end while

While a linear model oversimplifies a system's dynamics, it can be avoided in practice. To do so, we define the state space as an $\mathcal{HZ}$, which allows us to set boundaries, as the $\mathcal{HZ}$ is the collection of multiple constrained zonotopes that are, by definition, bounded. Therefore, if we consider for example a vehicle, its velocity can be restricted as needed in different regions. This will allow us to emulate the behaviour of a piecewise-linear model.

A notable characteristic of this approach is how it restricts the BRS within the state space. Typically, the computation of reachable sets utilizing the commonly known formulas in [2, Chapter 11] for the successor and predecessor sets includes computing all possible combinations of next or previous states and then intersect those with the state space to ensure that the reachable set is always contained within the state space. Formula (9), directly integrates this behavior within it, and there is no need for any additional computations to achieve that.

Example 4.1: For a more intuitive understanding consider the portion $\varphi_N = (true\,\underline{U}\,p_1)$ of the LTL (7). A high-level description of using this approach to construct this specific TLT is laid out in Algorithm 1.

The algorithm takes as input the LTL, the augmented state space, the target space, as well as the linear system's dynamics and outputs the root set node of the TLT, which in this case is the infinite horizon BRS. The hybrid zonotopes $\hat{\mathbb{S}}$ and $\mathbb{P}_1$ in this case correspond to the set of states satisfying the *true* and p_1 atomic propositions respectively. The core of

the algorithm involves the computation of this infinite horizon BRS starting from the target set. The process involves computing and combining predecessor sets of predecessor sets originating from the target $\mathbb{P}_1$ until it finally converges (no new states are added), which means the infinite horizon BRS has been computed. Finally, as shown in [4], the successful construction of the TLT tells which states can satisfy the specification.

V. SIMULATED PARKING SCENARIO

A parking mission typically entails the navigation of vehicles within non-trivial environments, often resulting in non-convex and disjoint sets. Since our work focuses on demonstrating the method's capability to effectively address such spatial complexities, the parking scenario is a fitting case study to demonstrate this work.

A. Vehicle Model

We define the following a discrete-time linear dynamics model to describe the behavior of a vehicle

$$\boldsymbol{x}_{t+1} = \underbrace{\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}}_{\boldsymbol{x}_t} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & dt \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} a_x \\ a_y \end{bmatrix}}_{\boldsymbol{u}_t}. \tag{10}$$

The state vector describes the vehicle's position and velocity in the $x-y$ plane, while the input vector is its acceleration. For this work we use a sampling time of $dt = 0.1s$.

B. Parking Environment

All simulations in this work are conducted within the environment in Figure 3. The linear model in (10) does not capture the the vehicle's orientation and velocity constraints. To compensate for that, one can encode such information within the state and input spaces themselves.

Consider, for example, the top and bottom horizontal lanes in a sub-figure of Figure 3; the velocity and acceleration constraints, as well as the traffic direction rules, can be described by augmenting the state space with the input space as $\hat{\boldsymbol{x}}_k = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{u}_k^T \end{bmatrix}^T$ and then using the following $\mathcal{HZ} = \{\mathbf{0}, G^c, G^b, \mathbf{0}, \mathbf{0}, \mathbf{0}\}$, where $G^c = \textit{diag(1.4, 0.05, 0.5, 1, 0.1, 0.1)}$, $G^b = [0\ 0.9\ 0.5\ 0\ 0\ 0]^T$.

The continuous generators of the $\mathcal{HZ}$ form a hyperrectangle whose size in each dimension is double the corresponding diagonal element. In other words, a road lane with a size of $(2.8 \times 0.1)m^2$ is formed. In that road section the velocities v_x, v_y are constrained in $([-0.5, 0.5],\ [-1.0, 1.0])\ m/s$ respectively, and the acceleration in $[-0.1, 0.1]\ m/s^2$. To duplicate this road as well as adjust its position and velocity parameters we use its binary generators. Since there is one binary generator, the $\mathcal{HZ}$ is equivalent to the union of these two constrained zonotopes $\mathcal{CZ}_1 = \{G^b, G^c, \mathbf{0}, \mathbf{0}\},\ \mathcal{CZ}_2 = \{-G^b, G_2, \mathbf{0}, \mathbf{0}\}$.

Each constrained zonotope represents one of the two horizontal lanes and forces the vehicle to abide by the traffic direction rules. Following similar reasoning, the rest of the augmented state space can be defined and collectively described using a single $\mathcal{HZ}$ and is denoted as $\hat{\mathbb{S}}$.

C. Constructing the TLTs

The case study considered in this work is to provide guarantees for an autonomous vehicle parking task. This mission was introduced in Example 2.1. Following that example, the resulting LTL specification is restated here.

$$\begin{gathered} \varphi = (\Diamond p_1) \vee (\Diamond p_2) \vee (\Diamond p_3) \vee (\Diamond p_4) \vee (\Diamond p_5), \\ p_i = \{\boldsymbol{x} \in \mathbb{S} | \boldsymbol{x} \in \mathbb{P}_i\}, \quad i = \{1, 2, ..., 5\}. \end{gathered} \tag{11}$$

An equivalent TLT capturing the formula (11) is shown in Figure 2. The construction of this TLT requires the computation of BRSs, as well as their union. Starting with a single branch for parking spot $\mathbb{P}_i$, $\mathcal{R}(\mathbb{S}, \mathbb{P}_i)$ needs to be computed in order to find all (x, y, v_x, v_y) states for which the vehicle can reach that parking spot. Using equation (9), we need to augment the state space to include the car's acceleration and thus use the augmented state space $\hat{\mathbb{S}}$ instead.

Given that the environment is static, the state space can be directly defined such that it only contains the roads and excludes any obstacles. In addition, to account for the vehicle's dimensionality, the state space is conservatively shrunk towards the center of each lane, as the reachable set computations assume a point mass object.

VI. RESULTS

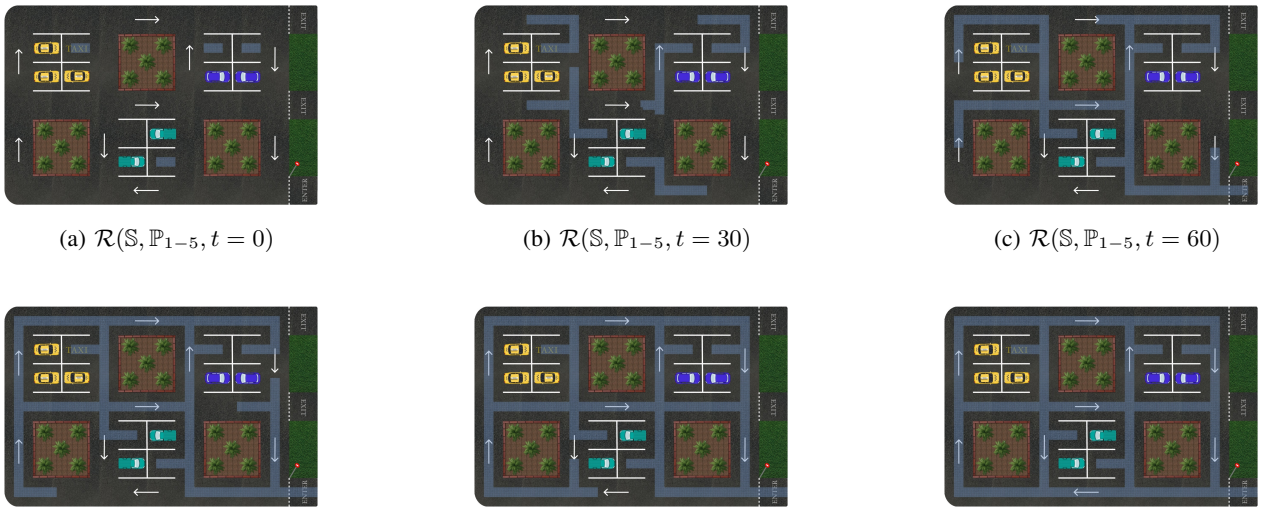
To assess the effectiveness of the combination of $\mathcal{HZ}$ reachability analysis and temporal logic, we now proceed to compute one by one all the set nodes of Figure 2.

Starting with the set nodes $\mathbb{P}_{1-5} \coloneqq \{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_5\}$, these represent the targets of our LTL; that is the five available parking spots. The set nodes $\mathcal{R}(\mathbb{S}, \mathbb{P}_1), \mathcal{R}(\mathbb{S}, \mathbb{P}_2), ..., \mathcal{R}(\mathbb{S}, \mathbb{P}_5)$ correspond to the BRS for each available parking spot and can be computed independently from each other in parallel to immediately generate the root set node of the tree.

The evolution of the computation of the root node is shown in Figure 3. There, we see its evolution in increments of 30 steps starting from the original target sets in Figure 3a until the set finally converges after 150 steps to the one shown in Figure 3f. For visualization purposes, the blue curve designates all positions (x, y) for which there is at least one velocity state that can satisfy the goal.

Right from the first time instance, we see how the $\mathcal{HZ}$ is capable of handling the five disjoint target sets simultaneously. The ability of the $\mathcal{HZ}$ to handle complex geometric representations is further verified during the next instances, where, for example, Figure 3b shows how the BRS, starting from the fourth parking spot, splits into two distinct paths. Then, we see how they all merge and grow together to cover the rest of the space. As expected, the final BRS covers the entire state space, aside the two exits of the parking lot, as a car is not allowed to move opposite to the traffic direction.

Earlier, we claimed that the simplicity of the dynamics model is not necessarily of great significance in describing complex tasks because the computation of BRSs using



(a) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 0)$ (b) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 30)$ (c) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 60)$

(d) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 90)$ (e) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 120)$ (f) $\mathcal{R}(\mathbb{S}, \mathbb{P}_{1-5}, t = 150)$

Fig. 3: Evolution of BRS from parking spots

$\mathcal{HZ}$s allows us to capture the input space information by augmenting the state space and thus enforcing the speed limit and traffic direction rules. This claim is indeed supported by the results since the evolution of the BRS always goes against the traffic direction, which is expected since we are computing a BRS and propagating backwards in time.

Sample computational times for the construction of the TLT both for the 4D and also for a simplified 2D version are in the range of $8.56s$ and $3.94s$, respectively. These results serve as a preliminary indication of the efficiency potential of $\mathcal{HZ}$s.

VII. CONCLUSION

Ensuring the safety of autonomous vehicles throughout their entire operation poses a challenging yet vital task that requires our attention. In this work, we have shown how LTL can be used to formally specify parking and navigation missions in dynamic environments and how one might use $\mathcal{HZ}$ reachability analysis to guarantee safety during these missions.

Although most works employing zonotope reachability analysis use them for their notable computational efficiency, few works address non-trivial spaces. This higher level of complexity in the problem is enabled through the use of temporal logic formalism, and the use of $\mathcal{HZ}$s allows us to accelerate the computation of the reachable sets.

Extensions of this work include applying this method to more abstract goals, such as high-level planning in smart cities, and exploiting the hybrid nature of this work through distributed system problems. Other interesting avenues are the extension of this approach to deal with dynamic environments and temporal tasks utilizing the potential of $\mathcal{HZ}$s for an online solution. Finally, it is important to perform formal control synthesis using the computed TLT in this work.

REFERENCES

[1] C. Baier and J. P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[2] F. Borelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[3] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, pp. 287–297, 2008.

[4] 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*, pp. 5071–5086, 2021.

[5] F. J. Jiang, Y. Gao, L. Xie, and K. H. Johansson, "Ensuring safety for vehicle parking tasks using hamilton-jacobi reachability analysis," in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1416–1421.

[6] 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, pp. 333–358, 2018.

[7] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European control conference (ECC)*. IEEE, 2013, pp. 502–510.

[8] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*, 2005, pp. 291–305.

[9] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, pp. 126–136, 2016.

[10] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2020.

[11] ——, "Constrained polynomial zonotopes," *Acta Informatica*, vol. 60, pp. 279—-316, 2023.

[12] T. J. Bird, H. C. Pangborn, N. Jain, and J. P. Koeln, "Hybrid zonotopes: A new set representation for reachability analysis of mixed logical dynamical systems," *Automatica*, 2023.

[13] T. J. Bird and N. Jain, "Unions and complements of hybrid zonotopes," *IEEE Control Systems Letters*, pp. 1778–1783, 2021.

[14] Y. Zhang, H. Zhang, and X. Xu, "Backward reachability analysis of neural feedback systems using hybrid zonotopes," *IEEE Control Systems Letters*, pp. 2779–2784, 2023.