Temporal Logic Trees for Model Checking and Control Synthesis of Uncertain Discrete-Time Systems

Yulong Gao, Alessandro Abate, *Senior Member, IEEE*, Frank J. Jiang, Mirco Giacobbe, Lihua Xie, *Fellow, IEEE*, and Karl Henrik Johansson, *Fellow, IEEE*

***Abstract*—We propose algorithms for performing model checking and control synthesis for discrete-time uncertain systems under linear temporal logic (LTL) specifications. We construct temporal logic trees (TLTs) from LTL formulae via reachability analysis. In contrast to automaton-based methods, the construction of the TLT is abstraction-free for infinite systems; that is, we do not construct discrete abstractions of the infinite systems. Moreover, for a given transition system and an LTL formula, we prove that there exist both a universal TLT and an existential TLT via minimal and maximal reachability analysis, respectively. We show that the universal TLT is an underapproximation for the LTL formula and the existential TLT is an overapproximation. We provide sufficient conditions and necessary conditions to verify whether a transition system satisfies an LTL formula by using the TLT approximations. As a major contribution of this work, for a controlled transition system and an LTL formula, we prove that a controlled TLT can be constructed from the LTL formula via a control-dependent reachability analysis. Based on the controlled TLT, we design an online control synthesis algorithm, under which a set of feasible control inputs can be generated at each time step. We also prove that this algorithm is recursively feasible. We illustrate the proposed methods for both finite and infinite systems and highlight the generality and online scalability with two simulated examples.**

***Index Terms*—Control synthesis, linear temporal logic, model checking, temporal logic trees.**

Manuscript received 2 July 2020; revised 29 April 2021; accepted 17 September 2021. Date of publication 6 October 2021; date of current version 27 September 2022. This work was supported in part by the Knut and Alice Wallenberg Foundation, the Swedish Strategic Research Foundation, the Swedish Research Council, the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, the HICLASS Project under Grant 113213, a partnership between the Aerospace Technology Institute (ATI), Department for Business, Energy and Industrial Strategy (BEIS) and Innovate U.K., and in part by the National Research Foundation, Singapore under its Medium Sized Center for Advanced Robotics Technology Innovation. Recommended by Associate Editor L. Palopoli. *(Corresponding author: Yulong Gao.)*

Yulong Gao is with the Division of Decision and Control Systems, KTH Royal Institute of Technology, 10044 Stockholm, Sweden, and also with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: yulongg@kth.se).

Alessandro Abate is with the Department of Computer Science, University of Oxford, OX13QD Oxford, U.K. (e-mail: aabate@cs.ox.ac.uk).

Frank J. Jiang is with the Division of Decision and Control Systems, KTH Royal Institute of Technology, 10044 Stockholm, Sweden (e-mail: frankji@kth.se).

Mirco Giacobbe is with the School of Computer Science, University of Birmingham, B152TT Birmingham, U.K. (e-mail: mirco.giacobbe@cs.ox.ac.uk).

Lihua Xie is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: elhxie@ntu.edu.sg).

Karl Henrik Johansson is with the Division of Decision and Control Systems, KTH Royal Institute of Technology, 10044 Stockholm, Sweden, and also with Digital Futures, 10044 Stockholm, Sweden (e-mail: kallej@kth.se).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2021.3118335.

Digital Object Identifier 10.1109/TAC.2021.3118335

I. Introduction

In THE recent past, the integration of computer science and control theory has promoted the development of new areas such as embedded systems design, hybrid systems theory, and, more recently, cyber–physical systems [1]–[4]. Given a model of a dynamical process and a specification (i.e., a description of desired properties), the following two fundamental problems arise:

1) *Model Checking:* automatically verifying whether the behavior of the model satisfies the given specification;
2) *Formal Control Synthesis:* automatically designing controllers (inputs to the system) so that the behavior of the model provably satisfies the given specification.

Both problems are of great interest in disparate and diverse applications, such as robotics, transportation systems, and safety-critical embedded system design. However, they are challenging problems when considering dynamical systems affected by uncertainty, and in particular uncertain infinite (uncountable) systems under complex, temporal logic specifications. In this article, we provide solutions to the model checking and formal control synthesis problems, for discrete-time uncertain systems under linear temporal logic (LTL) specification.

A. Related Work

In general, LTL formulae are expressive enough to capture many important properties, e.g., safety (nothing bad will ever happen), liveness (something good will eventually happen), and more complex combinations of Boolean and temporal statements [5].

In the area of formal verification, a dynamical process is by and large modeled as a finite transition system. A typical approach to both model checking and control synthesis for a finite transition system and an LTL formula leverages automata theory. It is known that each LTL formula can be transformed to an equivalent automaton [6]. The model checking problem can be solved by verifying whether the intersection of the trace set of the transition system and the set of accepted languages of the automaton expressing the negation of the LTL formula is empty or not [5]. The control synthesis problem can be solved by the following steps:

1) translate the LTL formula into a deterministic automaton;
2) build a "product automaton" between the transition system and the obtained automaton;
3) transform the product automaton into a game [7];
4) solve the game [8]–[10];
5) map the solution into a control strategy.

In recent years, the study of model checking and control synthesis for dynamical systems with continuous (uncountable) spaces, which extends the standard setup in formal verification, has attracted significant attention within the control community. This has enabled the formal control synthesis for interesting properties, which are more complex than the usual control objectives such as stability and set invariance. In order to adapt automaton-based methods to infinite systems, *abstraction* plays a central role in both model checking and control synthesis, which entails: 1) to abstract an infinite system to a finite transition system; 2) to conduct automaton-based model checking or control synthesis for the finite transition system; 3) if a solution is found, to map it back to the infinite system; otherwise, to refine the finite transition system and repeat the steps above.

In order to show the correctness of the solution obtained from the abstracted finite system over the infinite system, an equivalence or inclusion relation between the abstracted finite system and the infinite system needs to be established [11]. Relevant notions include (approximate) bisimulations and simulations. These relations and their variants have been explored for systems that are incrementally (input-to-state) stable [12], [13] or systems with similar properties [14]. Recent work [15] shows that the condition of approximate simulation can be relaxed to controlled globally asymptotic or practical stability with respect to a given set for nonlinear systems. We remark that such a condition holds for only a small class of systems in practice.

Based on abstractions, the problem of model checking for infinite systems has been studied in [16] and [17]. In [16], it is shown that model checking for discrete-time, controllable, linear systems from LTL formulae is decidable through an equivalent finite abstraction. In [17], the authors study the problem of verifying whether a linear system with additive uncertainty from some initial states satisfies a fragment of LTL formulae, which can be transformed to a deterministic Büchi automaton. The key idea is to use a formula-guided method to construct and refine a finite system abstracted from the linear system and guarantee their equivalence. Along the same line, the problem of control synthesis has also been widely studied for linear systems [18], nonlinear systems [19], stochastic systems [20], hybrid systems [21], and stochastic hybrid systems [22]. The applications of control synthesis under LTL specifications include single-robot control in dynamic environments [23], multi-robot control [24], and transportation control [25].

Beyond automata-based methods, alternative attempts have been made for specific model classes. Receding horizon methods are used to design controllers under LTL for deterministic linear systems [26] and uncertain linear systems [27]. The control of Markov decision processes under LTL is considered in [26] and further applied to multi-robot coordination in [28]. Control synthesis for dynamical systems has been extended also to other specifications like signal temporal logic (STL) [29] and probabilistic computational tree logic [30]. Interested readers may refer to the tutorial paper [31] and the book [32] for detailed discussions.

In this article, we propose a new tool for model checking and control synthesis that builds upon the close relationship between temporal logics and reachability analysis. In [33], the connection between STL and reachability analysis is studied, which inspires our work. Furthermore, reachability analysis on infinite systems [34], [35] and the computation of both forward and backward reachable sets [36]–[38] have been widely studied. As will be seen later in this article, the implementation of our methods is facilitated by these works.

B. Motivations

Although the last two decades have witnessed great progress on model checking and control synthesis for infinite systems from both theoretical and practical perspectives, there are some inherent restrictions in the dominant automaton-based methods.

First, abstraction from infinite systems to finite systems suffers from the curse of dimensionality: Abstraction techniques usually partition the state space, and transitions are constructed via reachability analysis. The computational complexity increases exponentially with the system dimension. Many works are dedicated to improving the computational efficiency by using overapproximation for (mixed) monotone systems [25] or by exploiting the structure of the uncertainty [22]. However, another issue with abstraction techniques is that only systems with "good properties" (e.g., incremental stability or smooth dynamics) might admit finite abstractions with guarantees, which limits their generality.

Second, there are few results for handling general LTL formulae when an infinite system comes with uncertainty (e.g., bounded disturbance or additive noise). In most contributions on control synthesis of uncertain systems, fragments of LTL formulae (e.g., bounded LTL or *co-safe* LTL) are usually taken into account [39], [40]. As mentioned before, the LTL formulae are defined over infinite trajectories, and it is difficult to control uncertainties propagating along such trajectories. This restriction results from conservative overapproximation in the computation of forward reachable sets, which is widely used for abstraction, and which leads to information loss when used with automaton-based methods.

Third, current methods usually lack online scalability. In many applications, full *a priori* knowledge of a specification cannot be obtained. For example, consider an automated vehicle required to move from some initial position to some destination without colliding into any obstacle (e.g., other vehicles and pedestrians). Since the trajectories of other vehicles and pedestrians cannot be accurately predicted, we cannot, in advance, define a specification that captures all the possibilities during the navigation process. Thus, offline design of automaton-based methods is significantly restricted.

Finally, the controller obtained from automaton-based methods usually only contains a single control policy. In some applications, e.g., human-in-the-loop control [41], [42], a set of feasible control inputs are needed to provide more degrees of freedom in the actual implementation. For example, [41] studies a control problem where humans are given a higher priority than the automated system in the decision making process. A controller is designed to provide a set of admissible control inputs with enough degrees of freedom to allow the human operator to easily complete the task.

C. Contributions

Motivated by the above, this article studies LTL model checking and reachability-based control synthesis for discrete-time uncertain systems. The main contributions of this article are threefold.

1) We construct tree structures from LTL formulae via reachability analysis over dynamical models. We denote the tree structure as a temporal logic tree (TLT). The connection between TLT and LTL is shown to hold for both uncertain finite and infinite models. The construction of the TLT is abstraction-free for infinite systems and admits online implementation, as demonstrated in Section VI. More specifically, given a system and an LTL formula, we prove that both a universal TLT and an existential TLT can be constructed for the LTL formula via minimal and maximal reachability analysis, respectively (Theorems III.1 and III.2). We also show that the universal TLT is an underapproximation for the LTL formula and the existential TLT is an overapproximation for the LTL formula. Our formulation does not restrict the generality of LTL formulae.

2) We provide a method for model checking of discrete-time dynamical systems using TLTs. We provide sufficient conditions to verify whether a transition system satisfies an LTL formula by using universal TLTs for underapproximating the satisfaction set or alternatively using existential TLTs for overapproximating the violation set (Theorem IV.1). Dually, we provide necessary conditions by using existential TLTs for overapproximating the satisfaction set or alternatively using universal TLTs for underapproximating the violation set (Theorem IV.2).

3) As a core and novel contribution of this article, we detail an approach for online control synthesis for a controlled transition system to guarantee that the controlled system will satisfy the specified LTL formula. Given a control system and an LTL formula, we construct a controlled TLT (Theorem V.1). Based on the obtained TLT, we design an online control synthesis algorithm, under which a set of feasible control inputs is generated at each time step (Algorithm 3). We prove that this algorithm is recursively feasible (Theorem V.2). We provide applications to show the scalability of our methods.

D. Organization

The remainder of the article is organized as follows. In Section II, we define the notion of transition system, recall the problem of reachability analysis, and provide preliminaries on LTL specifications. In Section III, we introduce TLT structures and show how to construct a TLT from a given LTL formula. In Section IV, we solve the LTL model checking problem through the constructed TLT. Section V solves the LTL control synthesis problem. In Section VI, we illustrate the effectiveness of our approaches with two numerical examples. In Section VII, we conclude the article with a discussion about our work and future directions.

Notation: Let $\mathbb{N}$ denote the set of nonnegative integers and $\mathbb{R}$ denote the set of real numbers. For some $q, s \in \mathbb{N}$, and $q < s$, let $\mathbb{N}_{\geq q}$ and $\mathbb{N}_{[q,s]}$ denote the sets $\{r \in \mathbb{N} \mid r \geq q\}$ and $\{r \in \mathbb{N} \mid q \leq r \leq s\}$, respectively. When $\leq, \geq, <$, and $>$ are applied to vectors, they are interpreted elementwise. For a matrix $A \in \mathbb{R}^{m\times n}$, A^T denotes its transpose.

II. Preliminaries

This section will first introduce transition systems and then recall reachability analysis and LTL.

A. Transition System

Definition II.1: A transition system TS is a tuple $\mathsf{TS} = (\mathbb{S}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ consisting of the following:

1) a set $\mathbb{S}$ of states;
2) a transition relation $\rightarrow \in \mathbb{S} \times \mathbb{S}$ [1];
3) a set $\mathbb{S}_0 \subseteq \mathbb{S}$ of initial states;
4) a set $\mathcal{AP}$ of atomic propositions;
5) a labeling function $L : \mathbb{S} \rightarrow 2^{\mathcal{AP}}$.

Definition II.2: A transition system TS is said to be finite if $|\mathbb{S}| < \infty$ and $|\mathcal{AP}| < \infty$.

Definition II.3: For $x \in \mathbb{S}$, the set $\mathsf{Post}(x)$ of direct successors of x is defined by $\mathsf{Post}(x) = \{x' \in \mathbb{S} \mid x \rightarrow x'\}$.

Definition II.4: A transition system TS is said to be nonblocking if $\mathsf{Post}(x) \neq \emptyset, \forall x \in \mathbb{S}$.

Definition II.5: A transition system TS is said to be deterministic if $|\mathbb{S}_0| = 1$ and $|\mathsf{Post}(x)| = 1, \forall x \in \mathbb{S}$.

Definition II.6: (Trajectory [2]) For a transition system TS, an infinite trajectory $\boldsymbol{p}$ starting from x_0 is a sequence of states $\boldsymbol{p} = x_0x_1 \ldots x_kx_{k+1} \ldots$ such that $\forall k \in \mathbb{N}$, $x_{k+1} \in \mathsf{Post}(x_k)$.

Through this article, we assume that the transition system TS is nonblocking, i.e., $\mathsf{Post}(x) \neq \emptyset, \forall x \in \mathbb{S}$. Denote by $\mathsf{Trajs}(x_0)$ the set of infinite trajectories starting from x_0. Let $\mathsf{Trajs}(\mathsf{TS}) = \cup_{x\in\mathbb{S}_0}\mathsf{Trajs}(x)$. For a trajectory $\boldsymbol{p}$, the kth state is denoted by $\boldsymbol{p}[k]$, i.e., $\boldsymbol{p}[k] = x_k$, the kth prefix is denoted by $\boldsymbol{p}[..k]$, i.e., $\boldsymbol{p}[..k] = x_0 \ldots x_k$, and the kth suffix is denoted by $\boldsymbol{p}[k..]$, i.e., $\boldsymbol{p}[k..] = x_kx_{k+1} \ldots$.

Example II.1: A traffic light can be red, green, yellow, or black (not working). The traffic light might stop working at any time. After it has been repaired, it turns red. Initially, the light is red. An illustration of such a traffic light is shown in Fig. 1(a). We can model the traffic light as a transition system $\mathsf{TS} = (\mathbb{S}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$:

[1]Here, the transition relation is not a functional relation, but instead for some state x, there may exist two different states x' and x'' such that $x \rightarrow x'$ and $x \rightarrow x''$ hold. For notational simplicity, we use $\rightarrow \in \mathbb{S} \times \mathbb{S}$, rather than $\rightarrow \in \mathbb{S} \times 2^{\mathbb{S}}$. The same claim holds for the controlled transition systems in Section V.

[2]Note that a trajectory $\boldsymbol{p} = x_0x_1 \ldots x_kx_{k+1} \ldots$ is different from a *trace*, which is the sequence of corresponding sets of atomic propositions, and is denoted by $L(x_0)L(x_1) \ldots L(x_k)L(x_{k+1}) \ldots$.

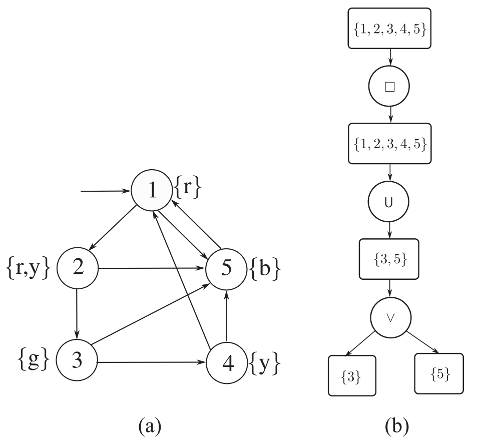


Fig. 1. (a) Transition system illustrating a traffic light example. Labels are shown aside the states. The initial state is denoted by an incoming edge. (b) TLT corresponding to an LTL formula $\varphi = \Box\Diamond(g \vee b)$ for this system. Note that $\Diamond\varphi = \text{true} \cup \varphi$.

1) $\mathbb{S} = \{1, 2, 3, 4, 5\}$;
2) $\rightarrow = \{(1, 2), (2, 3), (3, 4), (4, 1), (1, 5), (2, 5), (3, 5), (4, 5), (5, 1)\}$;
3) $\mathbb{S}_0 = \{1\}$;
4) $\mathcal{AP} = \{r, y, g, b\}$;
5) $L = \{1 \rightarrow \{r\}, 2 \rightarrow \{r, y\}, 3 \rightarrow \{g\}, 4 \rightarrow \{y\}, 5 \rightarrow \{b\}\}$. □

Remark II.1: We can rewrite the following discrete-time autonomous system as an infinite transition system:

$$\mathsf{S} : \begin{cases} x_{k+1} = f(x_k, w_k), \\ y_k = g(x_k) \end{cases}$$

where $x_k \in \mathbb{S}_\mathsf{S} \subseteq \mathbb{R}^{n_x}$, $w_k \in \mathbb{R}^{n_w}$, $y_k \in 2^{\mathcal{O}}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_x}$, and $g : \mathbb{R}^{n_x} \rightarrow 2^{\mathcal{O}}$. Here, $\mathcal{O}$ denotes the set of observations. At each time instant k, the disturbance w_k belongs to a compact set $\mathbb{W} \subset \mathbb{R}^{n_w}$. Denote by $\mathsf{Ini} \subseteq \mathbb{R}^{n_x}$ the set of initial states. The system S can be rewritten as an infinite transition system $\mathsf{TS}_\mathsf{S} = (\mathbb{S}_\mathsf{S}, \rightarrow, \mathsf{Ini}, \mathcal{O}, g)$, where $\forall x, x' \in \mathbb{S}_\mathsf{S}$, $x \rightarrow x'$ if and only if there exists $w \in \mathbb{W}$ such that $x' = f(x, w)$. □

B. Reachability Analysis

This subsection specifies the reachability analysis for a transition system TS. We first define the minimal reachable set and the maximal reachable set.

Definition II.7: Consider a transition system TS and two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. The k-step minimal reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^{\min}(\Omega_1, \Omega_2, k) = \{x_0 \in \mathbb{S} \mid \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0), \text{ s.t.}, \boldsymbol{p}[..k] = x_0 \dots x_k, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2\}.$$

The minimal reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^{\min}(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^{\min}(\Omega_1, \Omega_2, k).$$

The minimal reachable set can be recursively computed in the following way. For two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$, define

$$\mathbb{Q}^{\min}_{k+1} = \{x \in \Omega_1 \mid \mathsf{Post}(x) \subseteq \mathbb{Q}^{\min}_k\} \cup \mathbb{Q}^{\min}_k$$

with initialization $\mathbb{Q}^{\min}_0 = \Omega_2$. Then, $\mathcal{R}^{\min}(\Omega_1, \Omega_2) = \bigcup_{i \in \mathbb{N}} \mathbb{Q}^{\min}_k$.

Definition II.8: Consider a transition system TS and two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. The k-step maximal reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^{\max}(\Omega_1, \Omega_2, k) = \{x_0 \in \mathbb{S} \mid \exists \boldsymbol{p} \in \mathsf{Trajs}(x_0), \text{ s.t.}, \boldsymbol{p}[..k] = x_0 \dots x_k, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2\}.$$

The maximal reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^{\max}(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^{\max}(\Omega_1, \Omega_2, k).$$

Define

$$\mathbb{Q}^{\max}_{k+1} = \{x \in \Omega_1 \mid \mathsf{Post}(x) \cap \mathbb{Q}^{\max}_k \neq \emptyset\} \cup \mathbb{Q}^{\max}_k$$

with initialization $\mathbb{Q}^{\max}_0 = \Omega_2$. Similarly, $\mathcal{R}^{\max}(\Omega_1, \Omega_2) = \bigcup_{i \in \mathbb{N}} \mathbb{Q}^{\max}_k$.

We define the robust invariant set and the invariant set in the following.

Definition II.9: A set $\Omega_f \subseteq \mathbb{S}$ is said to be a robust invariant set of a transition system TS if for any $x \in \Omega_f$, $\mathsf{Post}(x) \subseteq \Omega_f$.

Definition II.10: For a set $\Omega \subseteq \mathbb{S}$, a set $\mathcal{RI}(\Omega) \subseteq \mathbb{S}$ is said to be the largest robust invariant set in Ω if each robust invariant set $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{RI}(\Omega)$.

Define

$$\mathbb{Q}^{\text{riv}}_{k+1} = \{x \in \mathbb{Q}^{\text{riv}}_k \mid \mathsf{Post}(x) \subseteq \mathbb{Q}^{\text{riv}}_k\}$$

with initialization $\mathbb{Q}^{\text{riv}}_0 = \Omega$. As shown in [43], the largest robust invariant set $\mathcal{RI}(\Omega) = \mathbb{Q}^{\text{riv}}_k$ for some $k \in \mathbb{N}$ if and only if $\mathbb{Q}^{\text{riv}}_k = \mathbb{Q}^{\text{riv}}_{k+1}$.

Definition II.11: A set $\Omega_f \subseteq \mathbb{S}$ is said to be an invariant set of a transition system TS if for any $x \in \Omega_f$, $\mathsf{Post}(x) \cap \Omega_f \neq \emptyset$.

Definition II.12: For a set $\Omega \subseteq \mathbb{S}$, a set $\mathcal{I}(\Omega) \subseteq \mathbb{S}$ is said to be the largest invariant set in Ω if each invariant set $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{I}(\Omega)$.

Define

$$\mathbb{Q}^{\text{iv}}_{k+1} = \{x \in \mathbb{Q}^{\text{iv}}_k \mid \mathsf{Post}(x) \cap \mathbb{Q}^{\text{iv}}_k \neq \emptyset\}$$

with initialization $\mathbb{Q}^{\text{iv}}_0 = \Omega$. Similarly, the largest invariant set $\mathcal{I}(\Omega) = \mathbb{Q}^{\text{iv}}_k$ for some $k \in \mathbb{N}$ if and only if $\mathbb{Q}^{\text{iv}}_k = \mathbb{Q}^{\text{iv}}_{k+1}$.

We can understand the reachable sets and invariant sets defined above as maps $\mathcal{R}^{\min} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \rightarrow 2^{\mathbb{S}}$, $\mathcal{R}^{\max} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \rightarrow 2^{\mathbb{S}}$, $\mathcal{RI} : 2^{\mathbb{S}} \rightarrow 2^{\mathbb{S}}$, and $\mathcal{I} : 2^{\mathbb{S}} \rightarrow 2^{\mathbb{S}}$, respectively. In the following, we will refer to them as "reachability operators."

C. LTL

An LTL formula is defined over a finite set of atomic propositions $\mathcal{AP}$ and both logic and temporal operators. The syntax of LTL can be described as

$$\varphi ::= \text{true} \mid a \in \mathcal{AP} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

where $\bigcirc$ and U denote the "next" and "until" operators, respectively. By using the negation and conjunction operators, we can define disjunction: $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. By employing the until operator, we can define: 1) eventually, $\Diamond\varphi = \text{true} \cup \varphi$; 2) always, $\Box\varphi = \neg\Diamond\neg\varphi$; and 3) weak-until, $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \Box\varphi_1$.

Definition II.13: (LTL semantics) For an LTL formula φ and a trajectory $\boldsymbol{p}$, the satisfaction relation $\boldsymbol{p} \vDash \varphi$ is defined as

$$\begin{aligned}
&\boldsymbol{p} \vDash a \Leftrightarrow a \in L(x_0),\\
&\boldsymbol{p} \vDash \neg\varphi \Leftrightarrow \boldsymbol{p} \nvDash \varphi,\\
&\boldsymbol{p} \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \wedge \boldsymbol{p} \vDash \varphi_2,\\
&\boldsymbol{p} \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \vee \boldsymbol{p} \vDash \varphi_2,\\
&\boldsymbol{p} \vDash \bigcirc\varphi \Leftrightarrow \boldsymbol{p}[1..] \vDash \varphi,\\
&\boldsymbol{p} \vDash \varphi_1 \mathsf{U} \varphi_2 \Leftrightarrow \exists j \in \mathbb{N} \text{ s.t.} \begin{cases} \boldsymbol{p}[j..] \vDash \varphi_2,\\ \forall i \in \mathbb{N}_{[0,j-1]}, \boldsymbol{p}[i..] \vDash \varphi_1 \end{cases}
\end{aligned}$$

where $a \in \mathcal{AP}$.

An LTL formula is in positive normal form if negations only occur adjacent to atomic propositions.

Lemma II.1: [5] For each LTL formula, there exists an equivalent LTL formula in weak-until positive normal form that is inductively defined as

$$\begin{aligned}
\varphi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi\\
\mid \varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1 \mathsf{W} \varphi_2.
\end{aligned}$$

Definition II.14: Consider a transition system TS and an LTL formula φ. The semantics of the universal form of φ, denoted by $\forall\varphi$, is

$$x_0 \vDash \forall\varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0), \boldsymbol{p} \vDash \varphi.$$

The semantics of the existential form of φ, denoted by $\exists\varphi$, is

$$x_0 \vDash \exists\varphi \Leftrightarrow \exists \boldsymbol{p} \in \mathsf{Trajs}(x_0), \boldsymbol{p} \vDash \varphi.$$

III. Temporal Logic Trees

This section will introduce the notion of TLT and establish a satisfaction relation between a trajectory and a TLT. Then, we construct TLTs from LTL formulae and discuss the approximation relation between them.

A. Definitions

A TLT refers to a tree that simulates a hierarchical structure with linked set nodes and operator nodes and serves as an alternative tool for model checking and control synthesis. The intuition of the TLT is that it collects a sequence of state sets (from root node to leaf nodes), indicating how a state trajectory ought to evolve from top to bottom in order to satisfy the corresponding temporal logic specification.

Definition III.1: A TLT is a tree for which the next holds:

1) each node is a *set* node, *a subset of* $\mathbb{S}$, or an *operator* node, *from* $\{\wedge, \vee, \bigcirc, \mathsf{U}, \Box\}$;
2) the root node and the leaf nodes are set nodes;
3) if a set node is not a leaf node, its unique child is an operator node;
4) the children of any operator node are set nodes.

Next we define the complete path and the minimal Boolean fragment for a TLT. Minimal Boolean fragments play an important role when simplifying the TLT for model checking and control synthesis in the following.

Definition III.2: A complete path of a TLT is a sequence of nodes and edges from the root node to a leaf node. Any subsequence of a complete path is called a fragment of the complete path.

Definition III.3: A minimal Boolean fragment of a complete path is one of the following fragments:

1) a fragment from the root node to the first Boolean operator node ($\wedge$ or $\vee$) in the complete path;
2) a segment from one Boolean operator node to the next Boolean operator node in the complete path;
3) a fragment from the last Boolean operator node of the complete path to the leaf node.

Example III.1: A TLT is shown in Fig. 1(b). We encode one of the complete paths of this TLT in the form of $\mathbb{X}_0 \Box \mathbb{X}_1 \mathsf{U} \mathbb{X}_2 \vee \mathbb{X}_3$, where $\mathbb{X}_0 = \mathbb{X}_1 = \{1, 2, 3, 4, 5\}$, $\mathbb{X}_2 = \{3, 5\}$, and $\mathbb{X}_3 = \{3\}$. For this complete path, the minimal Boolean fragments consist of $\mathbb{X}_0 \Box \mathbb{X}_1 \Diamond \mathbb{X}_2 \vee$ and $\vee \mathbb{X}_3$, which correspond to cases 1) and 3) in Definition III.3, respectively. $\Box$

We now define the satisfaction relation between a given trajectory and a complete path of a TLT.

Definition III.4: Consider a trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$ and encode a complete path of a TLT in the form of $\mathbb{X}_0 \odot_1 \mathbb{X}_1 \odot_2 \ldots \odot_{N_f} \mathbb{X}_{N_f}$ where N_f is the number of operators in the complete path, $\mathbb{X}_i \subseteq \mathbb{S}$ for all $i \in \mathbb{N}_{[0,N_f]}$ and $\odot_i \in \{\wedge, \vee, \bigcirc, \mathsf{U}, \Box\}$ for all $i \in \mathbb{N}_{[1,N_f]}$. The trajectory $\boldsymbol{p}$ is said to satisfy this complete path if $x_0 \in \mathbb{X}_0$ and there exists a sequence of time steps $k_0 k_1, \ldots, k_{N_f}$ with $k_i \in \mathbb{N}$ for all $i \in \mathbb{N}_{[0,N_f]}$ and $0 \triangleq k_0 \leq k_1 \leq k_2 \leq \ldots \leq k_{N_f}$ such that for all $i \in \mathbb{N}_{[0,N_f]}$, we have the following:

1) if $\odot_i = \wedge$ or $\odot_i = \vee$, $x_{k_i} \in \mathbb{X}_{i-1}$ and $x_{k_i} \in \mathbb{X}_i$;
2) if $\odot_i = \bigcirc$, $x_{k_i - 1} \in \mathbb{X}_{i-1}$ and $x_{k_i} \in \mathbb{X}_i$;
3) if $\odot_i = \mathsf{U}$, $x_j \in \mathbb{X}_{i-1}, \forall j \geq \mathbb{N}_{[k_{i-1}, k_i - 1]}$, and $x_{k_i} \in \mathbb{X}_i$;
4) if $\odot_i = \Box$, $x_j \in \mathbb{X}_i, \forall j \geq k_i$.

Consider a kth prefix $\boldsymbol{p}[..k] = x_0 x_1 \ldots x_k$ from $\boldsymbol{p}$ and a fragment from the complete path in the form of $\mathbb{X}_0 \odot_1 \mathbb{X}_1 \odot_2 \ldots \odot_{N_f'} \mathbb{X}_{N_f'}$ where $N_f' \leq N_f$. The prefix $\boldsymbol{p}[..k]$ is said to satisfy this fragment if $x_0 \in \mathbb{X}_0$, $x_k \in \mathbb{X}_{N_f'}$, and there exists a sequence of time steps $k_0 k_1, \ldots, k_{N_f'}$ with $k_i \in \mathbb{N}$ for all $i \in \mathbb{N}_{[0,N_f']}$ and $0 \triangleq k_0 \leq k_1 \leq k_2 \leq \ldots \leq k_{N_f'} \leq k$, such that for all $i \in \mathbb{N}_{[0,N_f']}$, 1)–3) holds and furthermore

4') if $\odot_i = \Box$, $x_j \in \mathbb{X}_i, \forall j \in \mathbb{N}_{[k_i,k]}$.

Definition III.5: A time coding of a TLT is an assignment of each operator node in the tree to a nonnegative integer.

The time coding is an extension of the time sequence in Definition III.4 to be associated with a tree structure and indicates when the operators in the TLT are activated along a given trajectory.

Definition III.6: Consider a trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$ and a TLT. The trajectory $\boldsymbol{p}$ is said to satisfy the TLT if there exists a time coding such that Algorithm 1 outputs $\mathtt{true}$.

Algorithm 1: TLT Satisfaction.

```
Input: a trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$, a TLT and a time coding
Output: true or false;
 1: construct a compressed tree via Algorithm 2 with input of the TLT;
 2: replace all set nodes of the compressed tree with false;
 3: for each complete path of the TLT do
 4:   if $\boldsymbol{p}$ satisfies the complete path under the time coding, then
 5:     set the corresponding leaf node in the compressed tree with true;
 6:   else
 7:     set the corresponding leaf node in the compressed tree with false;
 8:   end if
 9: end for
10: backtrack the tree;
11: return the root node of the tree.
```

Algorithm 1 provides a procedure to test if a trajectory satisfies a TLT under a given time coding. The TLT is first transformed into a compressed tree by removing all the temporal operators (lines 1–2), through Algorithm 2. Then, we check if the trajectory satisfies each complete path of the TLT under the time coding (lines 3–9). The outcome from lines 1–9 is a tree where the operator nodes are either $\vee$ or $\wedge$ and other nodes are either true or false. This is similar to the tree structure for the Boolean expressions. Finally, we backtrack the tree with output true or false (line 10). To backtrack the tree is to recursively evaluate all the subtrees from bottom to top with respect to the Boolean calculation. If the output is true, the trajectory satisfies the TLT; otherwise, the trajectory does not satisfy the TLT under the given time coding.

Algorithm 2 aims to obtain a tree in a compact form. Each minimal Boolean fragment is encoded according to Definition III.3. The notation $\odot_i$ denotes the operator node and N_f denotes the number of set nodes in the corresponding minimal Boolean fragment. We compress the sets in the minimal Boolean fragment to be a single set. The simplified tree consists of set nodes and Boolean operator nodes.

Example III.2: From Definition III.4, we can verify that the trajectory $\boldsymbol{p} = (1234)^\omega$ satisfies the complete path given in Example III.1 by choosing $k_0 = k_1 = 0$ and $k_2 = k_3 = 2$. It follows from Definition III.6 that this trajectory satisfies the corresponding TLT. □

B. Construction and Approximation of TLT

We define the approximation relations between TLTs and LTL formulae as follows.

Definition III.7: A TLT is said to be an underapproximation of an LTL formula φ if all the trajectories that satisfy the TLT also satisfy φ.

Definition III.8: A TLT is said to be an overapproximation of an LTL formula φ if all the trajectories that satisfy φ also satisfy the TLT.

Algorithm 2: Tree Compression.

```
Input: a tree
Output: a compressed tree
 1: for each complete path of the tree do
 2:   for each minimal Boolean fragment do
 3:     switchminimal Boolean fragmentdo
 4:     case1) in Definition III.3
 5:     encode the fragment in the form of $\mathbb{Y}_1 \odot_1 \ldots \odot_i \ldots \mathbb{Y}_{N_f} \odot_{N_f}$ with $\odot_{N_f} \in \{\wedge, \vee\}$;
 6:     replace the fragment with $\cup_{i=1}^{N_f} \mathbb{Y}_i \odot_{N_f}$;
 7:     case2) in Definition III.3
 8:     encode the fragment in the form of $\odot_1 \mathbb{Y}_1 \odot_2 \ldots \odot_{N_f} \mathbb{Y}_{N_f} \odot_{N_f+1}$ with $\odot_1, \odot_{N_f+1} \in \{\wedge, \vee\}$;
 9:     replace the fragment with $\odot_1 \cup_{i=1}^{N_f} \mathbb{Y}_i \odot_{N_f+1}$;
10:     case3) in Definition III.3
11:     encode the fragment in the form of $\odot_1 \mathbb{Y}_1 \odot_2 \ldots \odot_{N_f} \mathbb{Y}_{N_f}$ with $\odot_1 \in \{\wedge, \vee\}$;
12:     replace the fragment with $\odot_1 \cup_{i=1}^{N_f} \mathbb{Y}_i$;
13:     ▷ $\odot_i$ denotes the operator node and $N_f$ denotes the number of set nodes in the minimal Boolean fragment;
14:   end for
15: end for
16: return the updated tree.
```

The following two theorems show how to construct TLTs via reachability analysis for the LTL formulae and discuss their approximation relations.

Theorem III.1: For any transition system TS and any LTL formula φ, we have the following.

1) A TLT can be constructed from the formula $\forall\varphi$ through the reachability operators $\mathcal{R}^{\min}$ and $\mathcal{RI}$.
2) This TLT is an underapproximation of φ.

Proof: Here we provide a proof sketch. See Appendix A for a detailed proof.

We prove the constructability by the following three steps: 1) According to Lemma II.1, we transform the given LTL formula φ into an equivalent LTL formula in the weak-until positive normal form; 2) for each atomic proposition $a \in \mathcal{AP}$, we show that a TLT can be constructed from $\forall a$ (or $\forall\neg a$); 3) we leverage induction to show the following: given LTL formulae φ, φ_1, and φ_2 in weak-until positive normal form, if TLTs can be constructed from $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$, respectively, then TLTs can also be constructed through reachability operators $\mathcal{R}^{\min}$ and $\mathcal{RI}$ from the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall \bigcirc \varphi$, $\forall(\varphi_1 \mathsf{U} \varphi_2)$, and $\forall(\varphi_1 \mathsf{W} \varphi_2)$, respectively.

We follow a similar approach to prove an underapproximation relation between the constructed TLT and the LTL formula. The underapproximation occurs due to the conjunction operator and the presence of branching in the transition system. ■

Similarly, the following results hold.

Theorem III.2: For any transition system TS and any LTL formula φ, we have the following.

1) A TLT can be constructed from the formula $\exists\varphi$ through the reachability operators $\mathcal{R}^{\max}$ and $\mathcal{I}$.
2) This TLT is an overapproximation of φ.

Proof: The proof of the first part is similar to that of Theorem III.1 by replacing the universal quantifier $\forall$ and the reachability operators $\mathcal{R}^{\min}$ and $\mathcal{RI}$ with the existential quantifier $\exists$ and the operators $\mathcal{R}^{\max}$ and $\mathcal{I}$, respectively. Also, the proof of the second part is similar to that of Theorem III.1 by following the definitions of the maximal reachable set and invariant set.■

We call the constructed TLT of $\forall\varphi$ the *universal TLT* of φ and the TLT of $\exists\varphi$ the *existential TLT* of φ. We remark that the constructed TLT is not unique: This is because an LTL formula can have different equivalent expressions (e.g., normal forms). Despite this, the approximation relations between an LTL formula and the corresponding TLT still hold.

The following corollary shows that the approximation relation between TLTs and LTL formulae are tight for deterministic transition systems.

Corollary III.1: For any deterministic transition system TS and any LTL formula φ, the universal TLT and the existential TLT of φ are identical.

Proof: If the system is deterministic, it follows that for any $\Omega_1, \Omega_2 \subseteq \mathbb{S}$ and $\Omega \subseteq \mathbb{S}$, $\mathcal{R}^{\min}(\Omega_1, \Omega_2) = \mathcal{R}^{\max}(\Omega_1, \Omega_2)$ and $\mathcal{RI}(\Omega) = \mathcal{I}(\Omega)$. Then, by the same construction procedure, we have that the constructed universal TLT is the same as the constructed existential TLT. ■

Remark III.1: Computing reachable and invariant sets is a classical problem in the analysis of both dynamical systems and computer programs and plays a central role in the construction of the TLT. A discrete-time dynamical system can be seen as a while loop that repeats a difference equation; an invariant set can thus be seen as a loop invariant, as intended in Hoare logic. How to compute reachable and invariant sets is not the focus of this article. Interested readers are referred to relevant results [36]–[38] and associated computational tools, e.g., the multiparametric toolbox [44] and the Hamilton–Jacobi toolbox [45]. □

Example III.3: Consider the traffic light in Example II.1 and the LTL formula $\varphi = \Box\Diamond(g \vee b)$. We follow the proof of Theorem III.1 to show the correspondence between $\forall\varphi$ and the TLT in Fig. 1(b).

1) The universal TLT of g is a single set node, i.e., $\{3\}$, and the universal TLT of b is also a single set node, i.e., $\{5\}$.
2) The root node of the universal TLT of $g \vee b$ is the union of $\{3\}$ and $\{5\}$, i.e., $\{3, 5\}$.
3) The root node of the universal TLT of $\Diamond(g \vee b)$ is $\mathcal{R}^{\min}(\mathbb{S}, \{3, 5\}) = \{1, 2, 3, 4, 5\}$.
4) The root node of the universal TLT of $\Box\Diamond(g \vee b)$ is $\mathcal{RI}(\{1, 2, 3, 4, 5\}) = \{1, 2, 3, 4, 5\}$.

We can follow the same steps in the proof of Theorem III.2 to construct the existential TLT of φ, which is the same as the universal TLT of φ for the system in Example II.1. □

IV. Model Checking via TLT

This section focuses on the model checking problem.

Problem IV.1: Consider a transition system TS and an LTL formula φ. Verify whether $\mathsf{TS} \vDash \varphi$, i.e., $\forall x_0 \in \mathbb{S}_0$, $x_0 \vDash \forall\varphi$.

Thanks to the approximation relations between the TLTs and the LTL formulae, we obtain the following lemma.

Lemma IV.1: For any transition system TS and any LTL formula φ, we have the following.

1) $x_0 \vDash \forall\varphi$ if x_0 belongs to the root node of the universal TLT of φ.
2) $x_0 \vDash \exists\varphi$ only if x_0 belongs to the root node of the existential TLT of φ.

Proof: The first result follows from that the root node of the universal TLT is an underapproximation of the satisfaction set of φ, as shown in Theorem III.1. Dually, the second result follows from that the root node of the universal TLT is an overapproximation of the satisfaction set of φ, shown in Theorem III.2. ■

The next theorem provides two sufficient conditions for solving Problem IV.1.

Theorem IV.1: For a transition system TS and an LTL formula φ, $\mathsf{TS} \vDash \varphi$ *if* one of the following conditions holds.

1) The initial state set $\mathbb{S}_0$ is a subset of the root node of the universal TLT for φ.
2) No initial state from $\mathbb{S}_0$ belongs to the root node of the existential TLT for $\neg\varphi$.

Proof: Condition 1) directly follows from the first result of Lemma IV.1. Let us next prove condition 2). It follows that

$$\mathsf{TS} \vDash \varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(\mathsf{TS}), \boldsymbol{p} \vDash \varphi \Leftrightarrow \forall \boldsymbol{p} \in \mathsf{Trajs}(\mathsf{TS}), \boldsymbol{p} \nvDash \neg\varphi.$$

From the second result of Lemma IV.1, if x_0 does not belong to the root node of the existential TLT of $\neg\varphi$, we have $\boldsymbol{p} \nvDash \neg\varphi$, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0)$. Thus, condition 2) is sufficient for verifying $\mathsf{TS} \vDash \varphi$. ■

Similarly, we derive two necessary conditions for solving the model checking problem.

Theorem IV.2: For a transition system TS and an LTL formula φ, $\mathsf{TS} \vDash \varphi$ *only if* one of the following conditions holds.

1) The initial state set $\mathbb{S}_0$ is a subset of the root node of the existential TLT for φ.
2) No initial state from $\mathbb{S}_0$ belongs to the root node of the universal TLT for $\neg\varphi$.

Proof: Similar to Theorem IV.1. ■

Note that the approximation relations between the TLT and the LTL formula are tight for deterministic transition systems, as shown in Corollary III.1. In this case, the model checking problem can be tackled as follows.

Corollary IV.1: For a deterministic transition system TS and an LTL formula φ, $\mathsf{TS} \vDash \varphi$ if and only if the initial state set $\mathbb{S}_0$ is a subset of the root node of the universal (or existential) TLT for φ.

Proof: Follows from Corollary III.1. ■

The conditions in Theorems IV.1–IV.2 imply that one can directly do model checking by using TLTs, as shown in the following example.

Example IV.1: Let us continue to consider the traffic light and the LTL formula $\varphi = \Box\Diamond(g \vee b)$. Let us verify whether $\mathsf{TS} \vDash \varphi$ by using the above method. Since the unique initial state x_0 belongs to the root node of the universal TLT of φ shown in Fig. 1(b), it follows from condition 1) in Theorem IV.1 that $\mathsf{TS} \vDash \varphi$. Next, we show how to use condition 2) to verify that $\mathsf{TS} \vDash \varphi$.

First of all, we have $\neg\varphi = \Diamond\Box(\neg g \wedge \neg b)$. Following the proof of Theorem III.2, we construct the existential TLT of $\neg\varphi$.

1) The existential TLT of $\neg g$ is a single set node, i.e., $\{1, 2, 4, 5\}$ and the existential TLT of $\neg b$ is also a single set node, i.e., $\{1, 2, 3, 4\}$.
2) The root node of the existential TLT of $\neg g \wedge \neg b$ is the intersection of $\{1, 2, 4, 5\}$ and $\{1, 2, 3, 4\}$, i.e., $\{1, 2, 4\}$.
3) The root node of the existential TLT of $\Box(\neg g \wedge \neg b)$ is $\mathcal{I}(\{1, 2, 4\}) = \emptyset$.

As the existential TLT of $\neg\varphi$ is the empty set $\emptyset$, this implies that condition 2) in Theorem IV.1 holds, and, thus, $\mathsf{TS} \vDash \varphi$. $\square$

V. Control Synthesis via TLT

This section will show how to use the TLT to do control synthesis. Before that, we will introduce the notion of controlled transition system and recall the controlled reachability analysis.

A. Controlled Transition System

We first define a controlled transition system, which is used to model discrete-time uncertain control systems.

Definition V.1: A controlled transition system CTS is a tuple $\mathsf{CTS} = (\mathbb{S}, \mathbb{U}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ consisting of the following:

1) a set $\mathbb{S}$ of states;
2) a set $\mathbb{U}$ of control inputs;
3) a transition relation $\rightarrow \in \mathbb{S} \times \mathbb{U} \times \mathbb{S}$;
4) a set $\mathbb{S}_0$ of initial states;
5) a set $\mathcal{AP}$ of atomic propositions;
6) a labeling function $L : \mathbb{S} \rightarrow 2^{\mathcal{AP}}$.

Definition V.2: A controlled transition system CTS is said to be finite if $|\mathbb{S}| < \infty$, $|\mathbb{U}| < \infty$, and $|\mathcal{AP}| < \infty$.

Definition V.3: For $x \in \mathbb{S}$ and $u \in \mathbb{U}$, the set $\mathsf{Post}(x, u)$ of direct successors of x under u is defined by $\mathsf{Post}(x, u) = \{x' \in \mathbb{S} \mid x \xrightarrow{u} x'\}$.

Definition V.4: For $x \in \mathbb{S}$, the set $\mathbb{U}(x)$ of admissible control inputs at state x is defined by $\mathbb{U}(x) = \{u \in \mathbb{U} \mid \mathsf{Post}(x, u) \neq \emptyset\}$.

Definition V.5: (Policy) For a controlled transition system CTS, a *policy* $\boldsymbol{\mu} = \mu_0\mu_1 \ldots \mu_k \ldots$ is a sequence of maps $\mu_k : \mathbb{S}^{k+1} \rightarrow \mathbb{U}$. Denote by $\mathcal{M}$ the set of all policies.

Definition V.6: (Trajectory) For a controlled transition system CTS, an infinite *trajectory* $\boldsymbol{p}$ starting from x_0 under a policy $\boldsymbol{\mu} = \mu_0\mu_1 \ldots \mu_k \ldots$ is a sequence of states $\boldsymbol{p} = x_0x_1 \ldots x_k \ldots$ such that $\forall k \in \mathbb{N}$, $x_{k+1} \in \mathsf{Post}(x_k, \mu_k(\boldsymbol{p}[..k]))$. Denote by $\mathsf{Trajs}(x_0, \boldsymbol{\mu})$ the set of infinite trajectories starting from x_0 under $\boldsymbol{\mu}$.

We assume that the controlled transition system CS considered in this article is nonblocking, i.e., for each $x \in \mathbb{S}$, there exists an admissible control input $u \in \mathbb{U}$ such that $\mathsf{Post}(x, u) \neq \emptyset$.

Example V.1: A controlled transition system $\mathsf{CTS} = (\mathbb{S}, \mathbb{U}, \rightarrow, \mathbb{S}_0, \mathcal{AP}, L)$ is shown in Fig. 2(a), where

1) $\mathbb{S} = \{s_1, s_2, s_3, s_4\}$;
2) $\mathbb{U} = \{a_1, a_2\}$;
3) $\rightarrow = \{(s_1, a_1, s_2), (s_1, a_1, s_3), (s_2, a_1, s_2), (s_2, a_1, s_3), (s_2, a_1, s_3), (s_2, a_2, s_4), (s_3, a_1, s_2), (s_3, a_2, s_3), (s_4, a_1, s_2), (s_4, a_1, s_4)\}$;

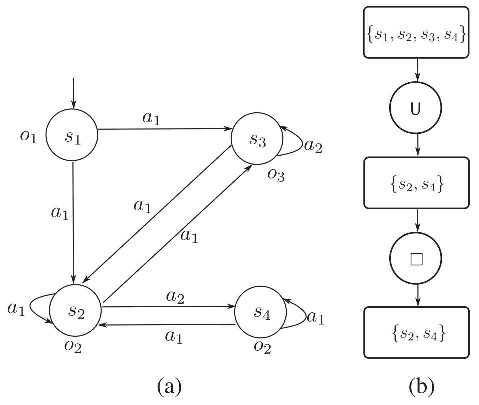


Fig. 2. (a) Graph description of a controlled transition system. (b) Controlled TLT of $\varphi = \Diamond\Box o_2$ for the system.

4) $\mathbb{S}_0 = \{s_1\}$;
5) $\mathcal{AP} = \{o_1, o_2, o_3\}$;
6) $L = \{s_1 \rightarrow \{o_1\}, s_2 \rightarrow \{o_2\}, s_3 \rightarrow \{o_3\}, s_4 \rightarrow \{o_2\}\}$. $\square$

Remark V.1: Similar to the transition systems, we can express the following discrete-time uncertain control system as an infinite controlled transition system:

$$\mathsf{CS} : \begin{cases} x_{k+1} = f(x_k, u_k, w_k), \\ y_k = g(x_k) \end{cases} \tag{1}$$

where $x_k \in \mathbb{S}_{\mathsf{CS}} \subseteq \mathbb{R}^{n_x}$ and $u_k \in \mathbb{U}_{\mathsf{CS}} \subset \mathbb{R}^{n_u}$, $w_k \in \mathbb{W} \subset \mathbb{R}^{n_w}$, $y_k \in 2^{\mathcal{O}}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_x}$, and $g : \mathbb{R}^{n_x} \rightarrow 2^{\mathcal{O}}$. Here, $\mathcal{O}$ denotes the set of observations. Assume that the sets $\mathbb{U}_{\mathsf{CS}}$ and $\mathbb{W}$ are compact. Let $\mathsf{Ini} \subseteq \mathbb{R}^{n_x}$ be the set of the initial states. Then, following the similar step in Remark II.1, the system CS can be rewritten as an infinite controlled transition system, $\mathsf{CTS}_{\mathsf{CS}} = (\mathbb{S}_{\mathsf{CS}}, \mathbb{U}_{\mathsf{CS}}, \rightarrow, \mathsf{Ini}, \mathcal{O}, g)$ where $\forall x, x' \in \mathbb{S}_{\mathsf{CS}}$ and $\forall u \in \mathbb{U}_{\mathsf{CS}}$, $x \xrightarrow{u} x'$ if and only if there exists $w \in \mathbb{W}$ such that $x' = f(x, u, w)$. $\square$

B. Controlled Reachability Analysis

This subsection will develop reachability analysis of a controlled transition system CTS.

Definition V.7: Consider a controlled transition system CTS and two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. The k-step controlled reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^c(\Omega_1, \Omega_2, k) = \{x_0 \in \mathbb{S} \mid \exists \boldsymbol{\mu} \in \mathcal{M} \text{ s.t., } \forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu}),$$
$$\boldsymbol{p}[..k] = x_0 \ldots x_k, \forall i \in \mathbb{N}_{[0,k-1]}, x_i \in \Omega_1, x_k \in \Omega_2\}.$$

The controlled reachable set from Ω_1 to Ω_2 is defined as

$$\mathcal{R}^c(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^c(\Omega_1, \Omega_2, k).$$

For two sets $\Omega_1, \Omega_2 \subseteq \mathbb{S}$, define

$$\mathbb{Q}^c_{k+1} = \{x \in \Omega_1 \mid \exists u \in \mathbb{U}(x), \mathsf{Post}(x, u) \subseteq \mathbb{Q}^c_k\} \cup \mathbb{Q}^c_k$$

with initialization $\mathbb{Q}^c_0 = \Omega_2$. Then, $\mathcal{R}^c(\Omega_1, \Omega_2) = \bigcup_{k \in \mathbb{N}} \mathbb{Q}^c_k$.

Definition V.8: A set $\Omega_f \subseteq \mathbb{S}$ is said to be a robust controlled invariant set (RCIS) of a transition system TS if for any $x \in \Omega_f$, there exists $u \in \mathbb{U}(x)$ such that $\mathsf{Post}(x, u) \subseteq \Omega_f$.

Definition V.9: For a set $\Omega \subseteq \mathbb{S}$, a set $\mathcal{RCI}(\Omega) \subseteq \mathbb{S}$ is said to be the largest RCIS in Ω if each RCIS $\Omega_f \subseteq \Omega$ satisfies $\Omega_f \subseteq \mathcal{RCI}(\Omega)$.

For a set $\Omega \subseteq \mathbb{S}$, define

$$\mathbb{Q}_{k+1}^{\text{rci}} = \{x \in \mathbb{Q}_k^{\text{rci}} \mid \exists u \in \mathbb{U}(x), \mathsf{Post}(x, u) \subseteq \mathbb{Q}_k^{\text{rci}}\}$$

with initialization $\mathbb{Q}_0^{\text{rci}} = \Omega$. As shown in [43], $\mathcal{RCI}(\Omega) = \mathbb{Q}_k^{\text{rci}}$ for some $k \in \mathbb{N}$ if and only if $\mathbb{Q}_k^{\text{rci}} = \mathbb{Q}_{k+1}^{\text{rci}}$.

The definitions of controlled reachable sets and RCISs provide us a way to synthesize the feasible control set, which is detailed in Algorithm 4. In the following, we treat the maps $\mathcal{R}^{\text{c}} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ and $\mathcal{RCI} : 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ as the reachability operators.

C. Construction and Approximation of TLT

The next theorem shows how to construct a TLT from an LTL formula for a controlled transition system and discusses its approximation relation.

Theorem V.1: For a controlled transition system CTS and any LTL formula φ, the following holds.

1) A TLT can be constructed from the formula φ through the reachability operators $\mathcal{R}^{\text{c}}$ and $\mathcal{RCI}$.
2) Given an initial state x_0, if there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the constructed TLT, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, then $\boldsymbol{p} \vDash \varphi$, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$.

Proof: The proof of the first part is similar to that of Theorem III.1 by replacing the reachability operators $\mathcal{R}^{\min}(\cdot)$ and $\mathcal{RI}(\cdot)$ with $\mathcal{R}^{\text{c}}(\cdot)$ and $\mathcal{RCI}(\cdot)$, respectively.

Similar to the underapproximation of the universal TLT in Theorem III.1, we can show that the path satisfying the constructed TLT in the first part also satisfies the LTL formula. Then, we can directly prove the second result. ■

Let us call the constructed TLT of φ in Theorem V.1 the controlled TLT of φ.

Remark V.2: Checking whether there exists a policy, such that all the resulting trajectories satisfy the obtained controlled TLT, is, in general, a hard problem. A straightforward necessary condition is that x_0 belongs to the root node of the controlled TLT; however, this is neither a necessary nor a sufficient condition on the existence of a policy such that all the resulting trajectories satisfy the given LTL formula. Instead, if we treat the controlled transition system as a nondeterministic transition system, a (rather conservative) necessary condition is that there exists at least one trajectory for this transition system that satisfies the LTL formula. This necessary condition can be further characterized by constructing the corresponding existential TLT and applying Theorem IV.2. □

Next, we will show how to construct the controlled TLT through an example.

Example V.2: Consider the controlled transition system in Example V.1. For an LTL formula $\varphi = \Diamond\Box o_2$, we can follow the steps in the proof of Theorem V.1 to construct the controlled TLT of φ, as shown in Fig. 2(b). □

D. Control Synthesis Algorithm

In this subsection, we solve the following control synthesis problem.

Problem V.1: Consider a controlled transition system CTS and an LTL formula φ. For an initial state $x_0 \in \mathbb{S}_0$, find, whenever existing, a sequence of feedback control inputs $\boldsymbol{u} = u_0 u_1 \ldots u_k \ldots$ such that the resulting trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$ satisfies φ.

Remark V.3: Note that the objective of the above problem is not to find a policy $\boldsymbol{\mu}$ but a sequence of control inputs that depend on the measured state. In general, synthesizing a policy $\boldsymbol{\mu}$ such that each trajectory $\boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$ satisfies φ is computationally intractable for infinite systems. Instead, we seek to find online a feasible control input at each time step, in a similar spirit to constrained control or receding horizon control. □

Instead of directly solving Problem V.1, we consider the following related task, whose solution is also a solution to Problem V.1, thanks to Theorem V.1.

Problem V.2: Consider a controlled transition system CTS and an LTL formula φ. For an initial state $x_0 \in \mathbb{S}_0$, find, whenever existing, a sequence of control inputs $\boldsymbol{u} = u_0 u_1 \ldots u_k \ldots$ such that the resulting trajectory $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$ satisfies the controlled TLT constructed from φ.

Algorithm 3 provides a solution to Problem V.2. In particular, Algorithm 3 presents an online feedback control synthesis procedure, which consists of three steps: 1) control tree—replace each set node of the TLT with a corresponding control set candidate (Algorithm 4); 2) compressed control tree—compress the control tree as a new tree whose set nodes are control sets and whose operator nodes are conjunctions and disjunctions (Algorithm 2); 3) backtrack on the control sets through a bottom-up traversal over the compressed control tree (Algorithm 5). If the output of Algorithm 3 is NExis, there does not exist a feasible solution to Problem V.2. We remark that Algorithm 3 is implemented in a similar way to receding horizon control with the prediction horizon being one.

Algorithm 4 aims to construct a control tree that enjoys the same tree structure as the input TLT. The difference is that all the set nodes are replaced with the corresponding control set nodes. The correspondence is established as follows:

1) whether the initial state x_0 belongs to the root node or not (lines 1–3);

2) whether the prefix $\boldsymbol{p}[..k]$ satisfies the fragment from the root node to the set node (lines 5–7);

3) whether or not the set node is a leaf node (lines 9–14);

4) which operator the child of the set node is (lines 16–23).

Algorithm 5 aims to backtrack a set by using the compressed tree. We update the parent of each Boolean operator node through a bottom-up traversal.

Note that the construction of a control tree in Algorithm 4 is closely related to the controlled reachability analysis in Section V-B. In lines 10–12, the computation of control set follows from the definition of RCIS. In lines 20–21, the definition of one-step controlled reachable set is used to compute the control set. In lines 22–23, the control set is synthesized from the definition of a controlled reachable set.

Algorithm 3: Online Feedback Control Synthesis via TLT.

```
Input: an initial state $x_0 \in \mathbb{S}_0$ and the controlled TLT of an LTL formula $\varphi$
Output: NExis or $(\boldsymbol{u}, \boldsymbol{p})$ with $\boldsymbol{u} = u_0 u_1 \ldots u_k \ldots$ and $\boldsymbol{p} = x_0 x_1 \ldots x_k \ldots$
1:  initialize $k = 0$;
2:  construct a control tree via Algorithm 4, with inputs $\boldsymbol{p}[..k] = x_0 \ldots x_k$ and the TLT;
3:  construct a compressed tree via Algorithm 2, with input the control tree;
4:  synthesize a control set $\mathbb{U}_k^{\varphi}$ via Algorithm 5, with input the compressed tree;
5:  if $\mathbb{U}_k^{\varphi} = \emptyset$ then
6:    stop and return NExis;
7:  else
8:    choose $u_k \in \mathbb{U}_k^{\varphi}$;
9:    implement $u_k$ and measure $x_{k+1} \in \mathsf{Post}(x_k, u_k)$;
10:   update $k = k + 1$ and go to Step 2;
11: end if
```

The following theorem shows that Algorithm 3 is recursively feasible. This means that initial feasibility implies future feasibility. This is an important property, particularly used in receding horizon control.

Theorem V.2: Consider a controlled transition system CTS, an LTL formula φ, and an initial state $x_0 \in \mathbb{S}_0$. Let x_0 and the controlled TLT of φ be the inputs of Algorithm 3. If there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, then we have the following.

1) The control set $\mathbb{U}_k^{\varphi}$ (line 8 of Algorithm 3) is nonempty for all $k \in \mathbb{N}$.
2) At each time step k, there exists at least one trajectory $\boldsymbol{p}$ with prefix $\boldsymbol{p}[..k+1] = x_0 \ldots x_k x_{k+1}$ under some policy such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall u_k \in \mathbb{U}_k^{\varphi}$ and $\forall x_{k+1} \in \mathsf{Post}(x_k, u_k)$.

Proof: The proof follows from the construction of the set $\mathbb{U}_k^{\varphi}$ in Algorithm 4 and the operations in Algorithms 2 and 5, and the definitions of controlled reachable sets and RCIS. If there exists a policy $\boldsymbol{\mu}$ such that $\boldsymbol{p}$ satisfies the controlled TLT of φ, $\forall \boldsymbol{p} \in \mathsf{Trajs}(x_0, \boldsymbol{\mu})$, we have that Algorithm 3 is feasible at each time step k, which implies that $\mathbb{U}_k^{\varphi} \neq \emptyset$. Furthermore, from Algorithm 4, each element in $\mathbb{U}_k^{\varphi}$ guarantees the one-step ahead feasibility for all realizations of $x_{k+1} \in \mathsf{Post}(x_k, u_k)$, which implies the result 2). ■

Theorem V.2 implies that if there exists a policy such that all the resulting trajectories satisfy the controlled TLT built from φ, then Algorithm 3 is always feasible at each time step in two senses: 1) the control set $\mathbb{U}_k^{\varphi}$ is nonempty; and 2) there always exists a feasible policy such that the trajectories with the realized prefix satisfy the controlled TLT.

Remark V.4: In Algorithm 3, the integration of Algorithms 2, 4, and 5 can be interpreted as a feedback control law. This control law is a set-valued map $\mathbb{S}^{k+1} \to 2^{\mathbb{U}}$ at time step k. Given the prefix $\boldsymbol{p}[..k] = x_0 \ldots x_k$, this map collects all the feasible control inputs such that the state can move along the TLT from $\boldsymbol{p}[..k]$. □

Algorithm 4: Control Tree.

```
Input: $\boldsymbol{p}[..k] = x_0 \ldots x_k$ and a TLT
Output: a control tree
1:  if $k = 0$ and $x_0 \notin$ the root node of TLT, then
2:    return $\emptyset$
3:  else
4:    for each set node $\mathbb{X}$ of TLT through a bottom-up traversal do
5:      if $\boldsymbol{p}[..k]$ does not satisfy the fragment from the root node to $\mathbb{X}$, then
6:        ▷ see Definition III.4;
7:        replace $\mathbb{X}$ with $\emptyset$;
8:      else
9:        if $\mathbb{X}$ is leaf node then
10:         if the parent of $\mathbb{X}$ is $\Box$ then
11:           replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathcal{RCI}(\mathbb{X})\}$;
12:         else
13:           replace $\mathbb{X}$ with $\mathbb{U}$;
14:         end if
15:       else
16:         switch the child of $\mathbb{X}$ do
17:         case $\wedge$ (or $\vee$)
18:         replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \cap_{i \in \mathrm{CH}} \mathbb{U}_{\mathrm{CH},i}$ (or $\mathbb{U}_{\mathbb{X}} = \cup_{i \in \mathrm{CH}} \mathbb{U}_{\mathrm{CH},i}$)
19:         ▷ for each Boolean operator node, CH collects its children and $\mathbb{U}_{\mathrm{CH},i}$ is the corresponding control set for each child;
20:         case $\bigcirc$
21:         replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}\}$; ▷ $\mathbb{Y}$ is the child of $\bigcirc$;
22:         case $\mathsf{U}$ or $\Box$
23:         replace $\mathbb{X}$ with $\mathbb{U}_{\mathbb{X}} = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{X}\}$;
24:       end if
25:     end if
26:   end for
27:   return the updated tree as the control tree.
28: end if
```

Remark V.5: Note that to implement Algorithm 3, we do not need to first check for the existence of a policy for the controlled TLT. The fact that a nonempty control set is synthesized by Algorithm 3 at each time step is necessary for the existence of the policy for the controlled TLT. We use the existence of the policy as a priori condition for proving the recursive feasibility of Algorithm 3 in Theorem V.1. □

Example V.3: Let us continue to consider the controlled transition system in Example V.1 and the LTL formula $\varphi = \Diamond\Box o_2$ in Example V.2. Implementing Algorithm 3, we obtain Table I. We can see that at each time step, we can synthesize a nonempty feedback control set. One realization is $s_1 \xrightarrow{a_1} s_3 \xrightarrow{a_2} s_3 \xrightarrow{a_1} s_2 \xrightarrow{a_1} s_3 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_4 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_4 \cdots$, of which the trajectory $\boldsymbol{p} = s_1 s_3 s_3 s_2 s_3 s_2 (s_4 s_2)^{\omega}$ satisfies both the controlled TLT and the formula φ.

Algorithm 5: Set Backtracking.

```
Input: a compressed tree
Output: a set $\mathbb{U}_k^{\varphi}$
1: for each Boolean operator node of the compressed tree
   through a bottom-up traversal do
2:   switch Boolean operator do
3:   case $\wedge$
4:   replace its parent with $\mathbb{Y}_{\mathrm{P}} \cup (\cap_{i\in \mathrm{CH}} \mathbb{Y}_{\mathrm{CH},i})$;
5:   case $\vee$
6:   replace its parent with $\mathbb{Y}_{\mathrm{P}} \cup (\cup_{i\in \mathrm{CH}} \mathbb{Y}_{\mathrm{CH},i})$;
7: end for
8: ▷ for each Boolean operator node, $\mathbb{Y}_{\mathrm{P}}$ denotes its
   parent, CH collects its children, and $\mathbb{Y}_{\mathrm{CH},i}$ is the
   corresponding control set for each child;
9: return the root node.
```

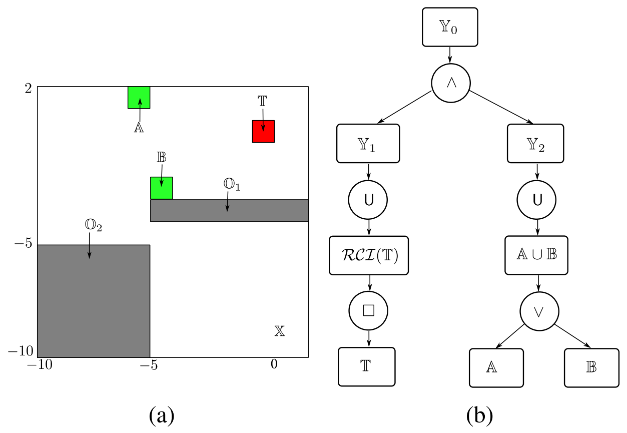


Fig. 3. (a) Scenario of Example 1. (b) Controlled TLT for the LTL formula $\varphi = ((a_1 \wedge \neg a_2 \wedge \neg a_3)\mathsf{U}\Box a_6) \wedge (\neg a_6 \mathsf{U}(a_4 \vee a_5))$ in Example 1, where $\mathbb{Y}_0 = \mathbb{Y}_1 \cap \mathbb{Y}_2$, $\mathbb{Y}_1 = \mathcal{R}^c(\mathbb{X}\setminus(\mathbb{O}_1\cup\mathbb{O}_2), \mathcal{RCI}(\mathbb{T}))$, and $\mathbb{Y}_2 = \mathcal{R}^c(\mathbb{X}\setminus\mathbb{T}, \mathbb{A}\cup\mathbb{B})$.

In this example, Algorithm 3 is recursively feasible since we can verify that the condition in Theorem V.2 holds. That is, there exists a policy such that all the resulting trajectories satisfy the controlled TLT: A feasible state-dependent stationary policy is $\boldsymbol{\mu} = \mu\mu\cdots$, where $\mu : \mathbb{S}\to\mathbb{U}$ with $\mu(s_1) = a_1, \mu(s_2) = a_2, \mu(s_3) = a_1$, and $\mu(s_4) = a_1$. Under this policy, there are two possible trajectories, $\boldsymbol{p} = s_1 s_3 (s_2 s_4)^{\omega}$ and $\boldsymbol{p} = s_1(s_2 s_4)^{\omega}$, both of which satisfy the controlled TLT and the LTL formula φ. □

VI. Examples

A. Obstacle Avoidance

Following the example of obstacle avoidance for double integrator in [46], we consider the following dynamical system:

$$x_{k+1} = \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} x_k + \begin{bmatrix}0.5\\ 1\end{bmatrix} u_k + w_k. \quad (2)$$

Different from [46], we may take into account the disturbance w_k that belongs to the disturbance set $\mathbb{W}$. We consider the same scenario as in [46], as shown in Fig. 3(a).

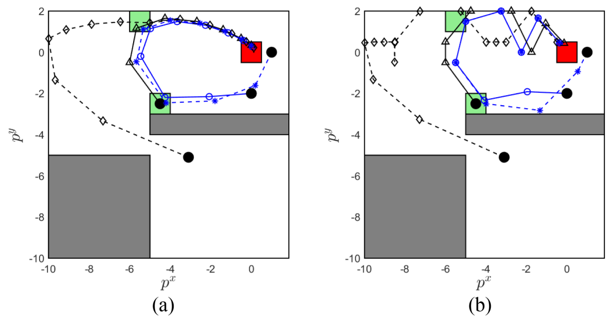


Fig. 4. State trajectories starting from different initial states for the co-safe LTL formula φ'. (a) Our approach. (b) Language-guided control synthesis [46].

The working space is $\mathbb{X} = \{z\in\mathbb{R}^2 \mid [-10,-10]^T \le z \le [2, 1.85]^T\}$ and the control constraint set is $\mathbb{U} = \{z \in \mathbb{R} \mid -2 \le z \le 2\}$. In Fig. 3(a), the obstacle regions are $\mathbb{O}_1 = \{z\in\mathbb{R}^2 \mid [-5,-4]^T \le z \le [1.85,-3]^T\}$ and $\mathbb{O}_2 = \{z\in\mathbb{R}^2 \mid [-10,-10]^T \le z \le [-5,-5]^T\}$, the target region is $\mathbb{T} = \{z \in \mathbb{R}^2 \mid [-0.5,-0.5]^T \le z \le [0.5, 0.5]^T\}$, and two visiting regions are $\mathbb{A} = \{z\in\mathbb{R}^2 \mid [-6,1]^T \le z \le [-5,2]^T\}$ and $\mathbb{B} = \{z\in\mathbb{R}^2 \mid [-5,-3]^T \le z \le [-4,-2]^T\}$.

Recall the system CS (1). Let the set of the observations be $\mathcal{O} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and, if $x\in\mathbb{X}$, we define the observation function as follows: if $x \in \mathbb{X}\cap\mathbb{O}_1$, $g(x) = \{a_1, a_2\}$; if $x\in\mathbb{X}\cap\mathbb{O}_2$, $g(x) = \{a_1, a_3\}$; if $x\in\mathbb{X}\cap\mathbb{A}$, $g(x) = \{a_1, a_4\}$; if $x\in\mathbb{X}\cap\mathbb{B}$, $g(x) = \{a_1,a_5\}$; if $x\in\mathbb{X}\cap\mathbb{T}$, $g(x) = \{a_1, a_6\}$; otherwise, $g(x) = \{a_1\}$. As shown in Remark V.1, we can rewrite the system (2) with the observation function as a controlled transition system with the set of atomic propositions $\mathcal{AP} = \mathcal{O}$ and the labeling function $L = g$.

We first compare our approach with the language-guided control synthesis in [46]. Consider the corresponding deterministic system, obtained setting $\mathbb{W} = \{0\}$. The specification is to visit the region $\mathbb{A}$ or region $\mathbb{B}$ and then the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$ and staying inside the working space $\mathbb{X}$. This specification can be expressed as a co-safe LTL formula $\varphi' = ((a_1\wedge\neg a_2\wedge\neg a_3)\mathsf{U}a_6)\wedge(\neg a_6\mathsf{U}(a_4 \vee a_5))$. Fig. 4 shows the state trajectories starting from different initial states for φ', generated using our approach and the approach in [46]. Let us introduce a performance function as $J_{\text{perf}} = \sum_{k=0}^{T_f}(\|u(k)\|^2)$, where T_f is the completion time. This can be used when implementing our approach: In view of the synthesized control sets $\mathbb{U}_k^{\varphi}$, control inputs u_k can be selected by solving a one-step optimization problem that minimizes $\|u(k)\|^2$, subject to $u(k)\in\mathbb{U}_k^{\varphi}$. As expected, the total J_{perf} of the trajectories in Fig. 4 under our approach is 35.95 and much less than the 70.52 obtained under the approach in [46]. Moreover, the TLT-based method in our article takes much shorter computation time: The construction of TLT using approximate reachability takes about 3 s, while the construction of control automaton in [46] takes about 70 s using the LanGuiCS tool[3]—the main reason is that the refinement in [46] is quite computationally expensive.

[3][Online]. Available: http://sites.bu.edu/hyness/languics/

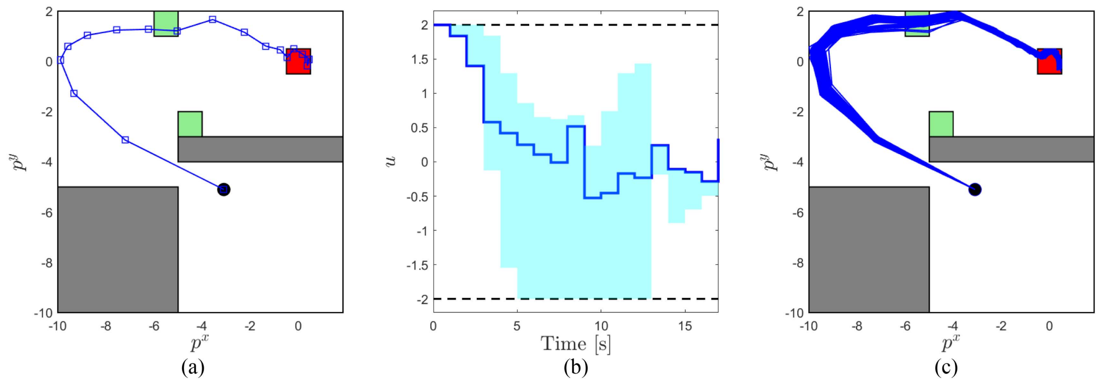


Fig. 5. Trajectories starting from the initial state $[-3.1, -5.1]^T$ for the non-co-safe LTL formula φ under our approach. (a) State trajectory x_k. (b) Control trajectory u_k (dashed line) together with control set $\mathbb{U}_k^\varphi$ (cyan region). (c) State trajectories for 100 realizations of disturbance trajectories.

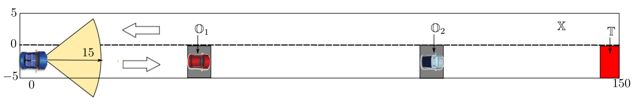


Fig. 6. Scenario illustration of Example 2: an automated vehicle plans to reach a target set $\mathbb{T}$ but with some unknown broken vehicles on the road.

We next show that our approach can further handle LTL formulae that are not co-safe as well as uncertain systems. Let the disturbance set $\mathbb{W} = \{z \in \mathbb{R}^2 \mid [-0.1, -0.1]^T \leq z \leq [0.1, 0.1]^T\}$. Here, we extend the co-safe LTL formula φ' to an LTL formula $\varphi = ((a_1 \wedge \neg a_2 \wedge \neg a_3)\mathsf{U}\Box a_6) \wedge (\neg a_6 \mathsf{U}(a_4 \vee a_5))$ that is not co-safe. The difference from φ' is to *always* stay inside the target region $\mathbb{T}$ after entering it. This specification cannot be handled by the approach in [46]. Under our approach instead, by computing inner approximations of the controlled reachable sets, we can construct the controlled TLT of φ and then use Algorithm 3 to synthesize controllers online. The constructed controlled TLT for φ is shown in Fig. 3(b). The state trajectories and the control trajectories are shown in Fig. 5 when the initial state is $[-3.1, -5.1]^T$. Fig. 5(a) and (c) shows the state trajectories for 1 and 100 realizations, respectively. The black dots denote the initial states. In this example, the target region $\mathbb{T}$ is an RCIS. After entering $\mathbb{T}$, the states stay there by using the controllers that ensure robust invariance. We can see that all state trajectories satisfy the required specification φ. The control information that corresponds to the state trajectory in Fig. 5(a) is shown in Fig. 5(b). Here, the dashed lines denote the control bounds, the cyan region represents the synthesized control sets $\mathbb{U}_k^\varphi$ in Algorithm 3, and the blue line represents the implemented control inputs u_k selected from the control sets $\mathbb{U}_k^\varphi$.

B. Online Specification Update

This example will show how the specification can be updated online when using our approach. As shown in Fig. 6, we consider a scenario where an automated vehicle plans to move to a target set $\mathbb{T}$ but with some unknown obstacles on the road. The sensing region of the vehicle is limited. We use a single integrator model with a sample period of 1 s to model the dynamics of the vehicle

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_k + w_k.$$

The working space is $\mathbb{X} = \{z \in \mathbb{R}^2 \mid [0, -5]^T \leq z \leq [150, 5]^T\}$, the control constraint set is $\mathbb{U} = \{z \in \mathbb{R}^2 \mid [-2, -0.5]^T \leq z \leq [2, 0.5]^T\}$, the disturbance set is $\mathbb{W} = \{z \in \mathbb{R}^2 \mid [-0.1, -0.1]^T \leq z \leq [0.1, 0.1]^T\}$, and the target region is $\mathbb{T} = \{z \in \mathbb{R}^2 \mid [145, -5]^T \leq z \leq [150, 0]^T\}$. We assume that $\mathbb{X}$, $\mathbb{U}$, and $\mathbb{W}$ are known *a priori* to the vehicle and the vehicle should move along the lane with the right direction unless lane change is necessary. In Fig. 6, there are two broken vehicles in the sets $\mathbb{O}_1 = \{z \in \mathbb{R}^2 \mid [40, -5]^T \leq z \leq [45, 0]^T\}$ and $\mathbb{O}_2 = \{z \in \mathbb{R}^2 \mid [100, -5]^T \leq z \leq [105, 0]^T\}$. We assume that $\mathbb{O}_1$ and $\mathbb{O}_2$ are unknown to the vehicle at the beginning. As long as the vehicle can sense them, they are known to the vehicle.

Let the initial state be $x_0 = [0.5, -2.5]^T$ and the sensing limitation is 15. At time step $k = 0$, the set of observations is $\mathcal{O} = \{a_1, a_2\}$ and if $x \in \mathbb{X}$, we define the observation function as follows: if $x \in \mathbb{X} \cap \mathbb{T}$, $g(x) = \{a_1, a_2\}$; otherwise, $g(x) = \{a_1\}$. The initial specification can be expressed as an LTL $\varphi = a_1 \mathsf{U} a_2$. By constructing the controlled TLT of φ shown in Fig. 7 and implementing Algorithm 3, we obtain one realization as shown in Fig. 8. We can see that the vehicle keeps moving straightforward until it senses the obstacle $\mathbb{O}_1$ at $[25.5, -2.4]^T$.

When the vehicle can sense $\mathbb{O}_1$, a new observation a_3 with $a_3 \neq a_1$ and $a_3 \neq a_2$ is added to the set $\mathcal{O}$, which becomes $\mathcal{O} = \{a_1, a_2, a_3\}$. If $x \in \mathbb{X}$, we update the observation function as follows: if $x \in \mathbb{X} \cap \mathbb{T}$, $g(x) = \{a_1, a_2\}$; if $x \in \mathbb{X} \cap \mathbb{O}_1$, $g(x) = \{a_1, a_3\}$; otherwise, $g(x) = \{a_1\}$. To avoid $\mathbb{O}_1$, the new specification is changed to be $\varphi' = \varphi \wedge (\Box \neg a_3)$. We can construct the TLT of φ' based on that of φ, which is shown in Fig. 7, and then continue to implement Algorithm 3. We can see that the vehicle changes lane from $[25.5, -2.4]^T$ and quickly merges back after overtaking $\mathbb{O}_1$. The trajectories are shown in Fig. 8. The vehicle is under the control with respect to φ' until it can sense $\mathbb{O}_2$ at $[86.3, -2.5]^T$.

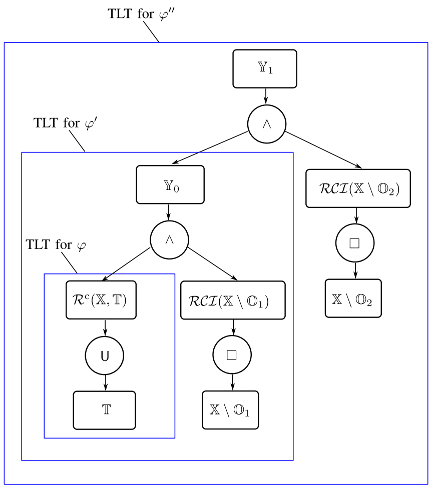

TLT for φ''
$\mathbb{Y}_1$
$\wedge$
TLT for φ'
$\mathbb{Y}_0$
$\mathcal{RCI}(\mathbb{X}\setminus\mathbb{O}_2)$
TLT for φ
$\wedge$
$\Box$
$\mathcal{R}^c(\mathbb{X},\mathbb{T})$
$\mathcal{RCI}(\mathbb{X}\setminus\mathbb{O}_1)$
$\mathbb{X}\setminus\mathbb{O}_2$
U
$\Box$
$\mathbb{T}$
$\mathbb{X}\setminus\mathbb{O}_1$

Fig. 7. Controlled TLT for the LTL formulae in Example 2, where $\varphi = a_1 \mathsf{U} a_2$, $\varphi' = \varphi \wedge (\Box\neg a_3)$, $\varphi'' = \varphi' \wedge (\Box\neg a_4)$, $\mathbb{Y}_0 = \mathcal{R}^c(\mathbb{X},\mathbb{T}) \cap \mathcal{RCI}(\mathbb{X}\setminus\mathbb{O}_1)$, and $\mathbb{Y}_1 = \mathcal{R}^c(\mathbb{X},\mathbb{T}) \cap \mathcal{RCI}(\mathbb{X}\setminus\mathbb{O}_1) \cap \mathcal{RCI}(\mathbb{X}\setminus\mathbb{O}_2)$.

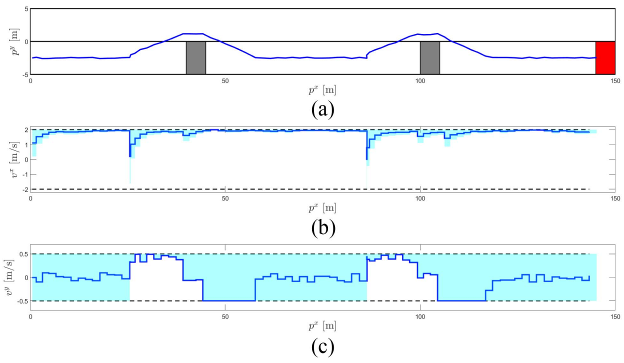


Fig. 8. Trajectories for one realization of disturbance trajectories. (a) State trajectories. (b) Control trajectories of x-axis. (c) Control trajectories of y-axis.

Similarly, when the vehicle can sense $\mathbb{O}_2$, we update $\mathcal{O} = \{a_1, a_2, a_3, a_4\}$ and the observation function as follows: if $x \in \mathbb{X} \cap \mathbb{T}$, $g(x) = \{a_1, a_2\}$; if $x \in \mathbb{X} \cap \mathbb{O}_1$, $g(x) = \{a_1, a_3\}$; if $x \in \mathbb{X} \cap \mathbb{O}_2$, $g(x) = \{a_1, a_4\}$; otherwise, $g(x) = \{a_1\}$. To avoid $\mathbb{O}_2$, the new specification is changed to be $\varphi'' = \varphi' \wedge (\Box\neg a_4)$. We can construct the TLT of φ'' based on that of φ', which is shown in Fig. 7, and then continue to implement Algorithm 3. We can see that the vehicle changes lane from $[86.3, -2.5]^T$ and quickly merges back after overtaking $\mathbb{O}_2$. Under the control with respect to φ'', the vehicle finally reaches the target set $\mathbb{T}$.

Fig. 8(a) shows the state trajectories, from which we can see that the whole specification is completed. Fig. 8(b) and (c) show the corresponding control inputs, where the dashed lines denote the control bounds. The cyan regions represent the synthesized control sets and the blue lines are the control trajectories. Furthermore, we repeat the above process for 100 realizations of the disturbance trajectories. The state trajectories for such 100 realizations are shown in Fig. 9.

We remark that, in this example, the control inputs are chosen to push the state to move down along the TLT as fast as possible.

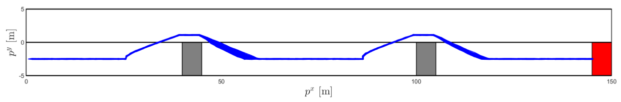


Fig. 9. State trajectories for 100 realizations of disturbance trajectories.

In detail, if the state x_k is the i-step reachable set in the set node $\mathcal{R}^c(\mathbb{X},\mathbb{T})$, we can generate a smaller control set from which the control input can push the state to the $(i-1)$-step reachable set. That is what we can see from Fig. 8, where almost all control inputs in the synthesized control sets along x-axis are positive.

VII. Conclusion

We have studied LTL model checking and control synthesis for discrete-time uncertain systems. Quite unlike automaton-based methods, our solutions build on the connection between LTL formulae and TLT structures via reachability analysis. For a transition system and an LTL formula, we have proved that the TLTs provide an underapproximation or overapproximation for the LTL via minimal and maximal reachability analysis, respectively. We have provided sufficient conditions and necessary conditions to the model checking problem. For a controlled transition system and an LTL formula, we have shown that the TLT is an underapproximation for the LTL formula and thereby proposed an online control synthesis algorithm, under which a set of feasible control inputs is generated at each time step. We have proved that this algorithm is recursively feasible. We have also illustrated the effectiveness of the proposed methods through several examples.

Future work includes the extension of TLTs to handle other general specifications (e.g., CTL*) and a broad experimental evaluation of our approach.

Appendix A
Proof of Theorem III.1

The whole proof is divided into two parts: The first part shows how to construct a TLT from the formula $\forall\varphi$ by means of the reachability operators $\mathcal{R}^m$ and $\mathcal{RI}$, while the second part shows that such TLT is an underapproximation for φ.

Construction: We follow three steps to construct a TLT.

Step 1: Rewrite the given LTL in the weak-until positive normal form. From Lemma II.1, each LTL formula has an equivalent LTL formula in the weak-until positive normal form, which can be inductively defined as

$$\varphi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi$$
$$\mid \varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1 \mathsf{W} \varphi_2.$$

Step 2: For each atomic proposition $a \in \mathcal{AP}$, construct the TLT with only a single set node from $\forall a$ or $\forall\neg a$. In detail, the set node for $\forall a$ is $L^{-1}(a) = \{x \in \mathbb{S} \mid a \in L(x)\}$, while the set node for $\forall\neg a$ is $\mathbb{S} \setminus L^{-1}(a)$. In addition, the TLT for $\forall$true (or $\forall$false) also has a single set node, which is $\mathbb{S}$ (or $\emptyset$).

Step 3: Based on Step 2, follow the induction rule to construct the TLT for any LTL formula in the weak-until positive normal

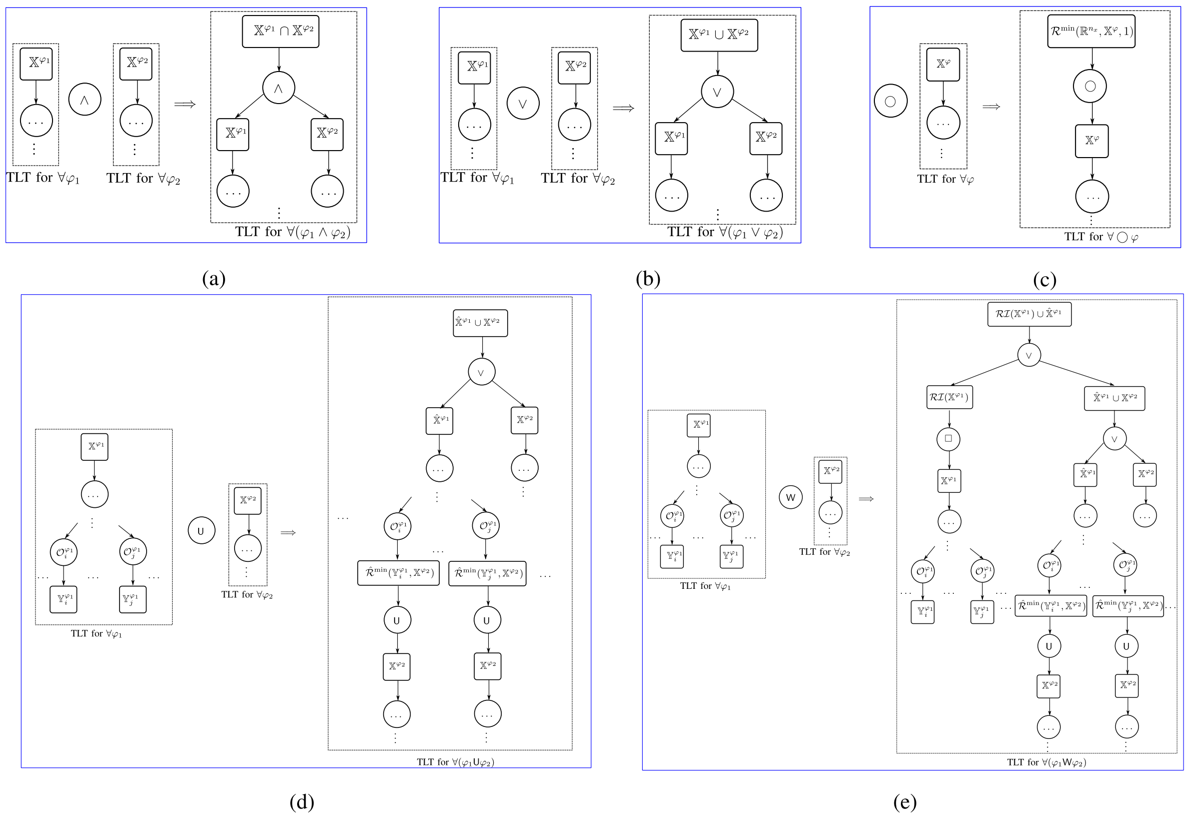

TLT for $\forall\varphi_1$
TLT for $\forall\varphi_2$
TLT for $\forall(\varphi_1 \wedge \varphi_2)$
TLT for $\forall(\varphi_1 \vee \varphi_2)$
TLT for $\forall\varphi$
TLT for $\forall\bigcirc\varphi$
TLT for $\forall(\varphi_1\mathsf{U}\varphi_2)$
TLT for $\forall(\varphi_1\mathsf{W}\varphi_2)$
(a)
(b)
(c)
(d)
(e)

Fig. 10. TLT construction. (a) $\forall(\varphi_1 \wedge \varphi_2)$. (b) $\forall(\varphi_1 \vee \varphi_2)$. (c) $\forall \bigcirc \varphi$. (d) $\forall(\varphi_1\mathsf{U}\varphi_2)$. (e) $\forall(\varphi_1\mathsf{W}\varphi_2)$. Here, the circles denote the operator nodes and the rectangles denote the set nodes.

TABLE I
ONLINE IMPLEMENTATION UNDER ALGORITHM 3

Time k	State x_k	Control set $\mathbb{U}_k^{\varphi}$	Control input u_k
0	s_1	$\{a_1\}$	a_1
1	s_3	$\{a_1, a_2\}$	a_2
2	s_3	$\{a_1, a_2\}$	a_1
3	s_2	$\{a_1, a_2\}$	a_1
4	s_3	$\{a_1, a_2\}$	a_1
5	s_2	$\{a_1, a_2\}$	a_2
6	s_4	$\{a_1\}$	a_1
7	s_2	$\{a_1, a_2\}$	a_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

form. More specifically, we will show that given the LTL formulae φ, φ_1, and φ_2 in the weak-until positive normal form, if the TLTs can be constructed from $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$, respectively, then the TLTs can be thereby constructed from the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall\bigcirc\varphi$, $\forall(\varphi_1\mathsf{U}\varphi_2)$, and $\forall(\varphi_1\mathsf{W}\varphi_2)$, respectively.

For $\forall(\varphi_1 \wedge \varphi_2)$ (or $\forall(\varphi_1 \vee \varphi_2)$), we construct the TLT by connecting the root nodes of the TLTs for $\forall\varphi_1$ and $\forall\varphi_2$ through the operator $\wedge$ (or $\vee$) and taking the intersection (or union) of two root nodes, as shown in Fig. 10(a) and (b). For $\forall\bigcirc\varphi$, we denote by $\mathbb{X}^{\varphi}$ the root node of the TLT for $\forall\varphi$ and then construct the TLT by adding a new set node $\mathcal{R}^{\min}(\mathbb{S}, \mathbb{X}^{\varphi}, 1)$ to be the parent of $\mathbb{X}^{\varphi}$ and connecting them through the operator $\bigcirc$, as shown in Fig. 10(c).

For $\forall(\varphi_1\mathsf{U}\varphi_2)$, the TLT construction is as follows. Denote by $\{(\mathbb{Y}_i^{\varphi_1}, \mathcal{O}_i^{\varphi_1})\}_{i=1}^{N^{\varphi_1}}$ all the pairs comprising a leaf node and its corresponding parent in the TLT of $\forall\varphi_1$, where N^{φ_1} is the number of the leaf nodes. Here, $\mathbb{Y}_i^{\varphi_1}$ is the ith leaf node and $\mathcal{O}_i^{\varphi_1}$ is its parent. Denote by $\mathbb{X}^{\varphi_2}$ the root node of TLT for $\forall\varphi_2$. We first change each leaf node $\mathbb{Y}_i^{\varphi_1}$ to $\hat{\mathcal{R}}^{\min}(\mathbb{Y}_i^{\varphi_1}, \mathbb{X}^{\varphi_2}) = (\mathcal{R}^{\min}(\mathbb{Y}_i^{\varphi_1}, \mathbb{X}^{\varphi_2}) \setminus \mathbb{X}^{\varphi_2}) \cup (\mathbb{Y}_i^{\varphi_1} \cap \mathbb{X}^{\varphi_2})$. We then update the new tree for $\forall\varphi_1$ from the leaf node to the root node according to the definition of the operators. After that, we take N^{φ_1} copies of the TLT of φ_2. We set the root node of each copy as the child of each new leaf node, respectively, and connect them through the operator U. Finally, we have one more copy of the TLT of $\forall\varphi_2$ and connect this copy and the new tree through the disjunction $\vee$. An illustrative diagram is given in Fig. 10(d).

For the fragment $\forall(\varphi_1\mathsf{W}\varphi_2)$, we first recall that $\varphi_1\mathsf{W}\varphi_2 = \varphi_1\mathsf{U}\varphi_2 \vee \Box\varphi_1$. Let $\varphi' = \varphi_1\mathsf{U}\varphi_2$ and $\varphi'' = \Box\varphi_1$. Denote by $\mathbb{X}^{\varphi_1}$ the root node of the TLT for $\forall\varphi_1$. We first construct the TLT of $\forall\varphi'$ as described above. Second, we further construct the TLT of $\forall\varphi''$ by adding a new node $\mathcal{RI}(\mathbb{X}^{\varphi_1})$ as the parent of $\mathbb{X}^{\varphi_1}$ and connecting them through $\Box$. Then, we construct the TLT of $\forall(\varphi' \vee \varphi'')$. An illustrative diagram is given in Fig. 10(e).

Underapproximation: First, it is very easy to verify that the constructed TLT above with a single set node $L^{-1}(a)$ (or $\mathbb{S} \setminus L^{-1}(a)$ or $\mathbb{S}$ or $\emptyset$) for $\forall a$ (or $\forall\neg a$ or $\forall$true or $\forall$false) is an underapproximation for $a \in \mathcal{AP}$ (or $\neg a$ or true or false) and the underapproximation relation in these cases is also tight.

Next we also follow the induction rule to show that the constructed TLT from $\forall\varphi$ is an underapproximation for φ. Consider LTL formulae φ, φ_1, and φ_2. We will show that if the constructed TLTs of $\forall\varphi$, $\forall\varphi_1$, and $\forall\varphi_2$ are the underapproximations of φ, φ_1, and φ_2, respectively, then the TLTs constructed above for the formulae $\forall(\varphi_1 \wedge \varphi_2)$, $\forall(\varphi_1 \vee \varphi_2)$, $\forall \bigcirc \varphi$, $\forall(\varphi_1 \mathsf{U} \varphi_2)$, and $\forall(\varphi_1 \mathsf{W} \varphi_2)$ are the underapproximations of $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\bigcirc\varphi$, $\varphi_1 \mathsf{U} \varphi_2$, and $\varphi_1 \mathsf{W} \varphi_2$, respectively.

According to the set operation (intersection or union) or the definition of one-step minimal reachable set, it is easy to verify that the constructed TLT for $\forall(\varphi_1 \wedge \varphi_2$ (or $\forall(\varphi_1 \vee \varphi_2)$ or $\forall \bigcirc \varphi$) is an underapproximation for $\varphi_1 \wedge \varphi_2$ (or $\varphi_1 \vee \varphi_2$ or $\bigcirc\varphi$) if the TLTs of $\forall\varphi_1$ and $\forall\varphi_2$, and $\forall\varphi$ are underapproximations for φ_1, φ_2, and φ, respectively.

Let us consider $\varphi_1 \mathsf{U} \varphi_2$. Assume that a trajectory $\boldsymbol{p}$ satisfies the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$. Recall the construction of the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$ from $\forall\varphi_1$ and $\forall\varphi_2$. According to the definition of minimal reachable set, we have 1) $\boldsymbol{p}$ satisfies the TLT of $\forall\varphi_2$ *or* 2) there exists that $j \in \mathbb{N}$ such that $\boldsymbol{p}[j..]$ satisfies the TLT of $\forall\varphi_2$ and for all $i \in \mathbb{N}_{[0,j-1]}$, the trajectory $p[i..]$ satisfies the TLT of $\forall\varphi_1$. Under the assumption that the TLTs of $\forall\varphi_1$ and $\forall\varphi_2$ are the underapproximations of φ_1 and φ_2, respectively, we have that there exists $j \in \mathbb{N}$ such that $\boldsymbol{p}[j..] \vDash \varphi_2$ and for all $i \in \mathbb{N}_{[0,j-1]}$, $p[i..] \vDash \varphi_1$, which implies that $\boldsymbol{p} \vDash \varphi_1 \mathsf{U} \varphi_2$. Thus, the TLT of $\forall(\varphi_1 \mathsf{U} \varphi_2)$ is an approximation of $\varphi_1 \mathsf{U} \varphi_2$.

Recall that $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \square\varphi_1$. Following the proofs for until operator U and the disjunction $\vee$ and the definition of the robust invariant set, it yields that the constructed TLT of $\forall(\varphi_1 \mathsf{W} \varphi_2)$ is an underapproximation of $\varphi_1 \mathsf{W} \varphi_2$.

The proof is complete. $\square$

Acknowledgment

The authors would like to thank Dr. Ren (Shanghai University) and H. Hasanbeig (University of Oxford) for helpful discussions and feedback.

References

[1] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Berlin, Germany: Springer, 2009.

[2] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Chelmsford, MA, USA: Courier Corporation, 2013.

[3] R. Alur, *Principles of Cyber-Physical Systems*. Cambridge, MA, USA: MIT Press, 2015.

[4] R. Alur, M. Giacobbe, T. A. Henzinger, K. G. Larsen, and M. Mikucionis, "Continuous-time models for system design and analysis," in *Proc. Comput. Softw. Sci., Ser. Lecture Notes Comput. Sci.*, 2019, pp. 452–477.

[5] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[6] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," in *Logics for Concurrency*, F. Moller and G. Birtwistle, Eds. Berlin, Germany: Springer, 1996, pp. 238–266.

[7] M. O. Rabin, "Decidability of second-order theories and automata on infinite trees," *Trans. Amer. Math. Soc.*, vol. 141, pp. 1–35, 1969.

[8] E. A. Emerson, "Automata, tableaux, and temporal logics," in *Proc. Workshop Log. Programs*, 1985, pp. 79–88.

[9] N. Piterman and A. Pnueli, "Faster solutions of Rabin and Streett games," in *Proc. 21st Annu. IEEE Symp. Logic Comput. Sci.*, 2006, pp. 275–284.

[10] F. Horn, "Streett games on finite graphs," in *Proc. 2nd Workshop Games Des. Verification*, 2005, pp. 1–12.

[11] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 782–798, May 2007.

[12] A. Girard, G. Pola, and G. J. Pappas, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 116–126, Jan. 2010.

[13] M. Zamani, P. M. Esfahani, R. Majumdar, A. Abate, and J. Lygeros, "Symbolic control of stochastic systems via approximately bisimilar finite abstractions," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3135–3150, Dec. 2014.

[14] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1804–1809, Jul. 2012.

[15] P. Yu and D. V. Dimarogonas, "Approximately symbolic models for a class of continuous-time nonlinear systems," in *Proc. 58th IEEE Conf. Decis. Control*, 2019, pp. 4349–4354.

[16] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2003, pp. 498–513.

[17] B. Yordanov, J. Tumová, I. Cerná, J. Barnat, and C. Belta, "Formal analysis of piecewise affine systems through formula guided refinement," *Automatica*, vol. 49, no. 1, pp. 261–266, 2013.

[18] B. Yordanov, J. Tumová, I. Cerná, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1491–1504, Jun. 2012.

[19] P.-J. Meyer and D. V. Dimarogonas, "Hierarchical decomposition of LTL synthesis problem for nonlinear control systems," *IEEE Trans. Autom. Control*, vol. 64, no. 11, pp. 4676–4683, Nov. 2019.

[20] S. Haesaert and S. Soudjani, "Robust dynamic programming for temporal logic control of stochastic systems," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2496–2511, Jun. 2020.

[21] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. 47th IEEE Conf. Decis. Control*, 2008, pp. 2117–2122.

[22] N. Cauchi and A. Abate, "StocHy-automated verification and synthesis of stochastic processes," in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2019, pp. 258–259.

[23] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1593–1607, 2014.

[24] M. Guo, J. Tumová, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3948–3962, Dec. 2016.

[25] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Control Netw. Syst.*, vol. 3, no. 2, pp. 162–171, Jun. 2016.

[26] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[27] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[28] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical LTL-task MDPs for multi-agent coordination through auctioning and learning," 2019. [Online]. Available: http://kth.diva-portal.org

[29] L. Lindemann and D. V. Dimarogonas, "Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks," *Automatica*, vol. 106, pp. 284–293, 2019.

[30] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd Int. Conf. Comput. Aided Verification*, 2011, pp. 585–591.

[31] C. Belta, "Formal synthesis of control strategies for dynamical systems," in *Proc. 55th IEEE Conf. Decis. Control*, 2016, pp. 3407–3431.

[32] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Berlin, Germany: Springer, 2017.

[33] M. Chen, Q. Tam, S. C. Livingston, and M. Pavone, "Signal temporal logic meets Hamilton-Jacobi reachability: Connections and applications," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2018. [Online]. Available: http://asl.stanford.edu/wp-content/papercite-data/pdf/Chen.Tam.Livingston.Pavone.WAFR18.pdf

[34] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Berlin, Germany: Springer, 2007.

[35] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[36] 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 Trans. Autom. Control*, vol. 63, no. 11, pp. 3675–3688, Nov. 2018.

[37] M. Althoff and B. H. Krogh, "Reachability analysis of nonlinear differential-algebraic systems," *IEEE Trans. Autom. Control*, vol. 59, no. 2, pp. 371–383, Feb. 2014.

[38] I. M. Mitchell, "Scalable calculation of reach sets and tubes for nonlinear systems with terminal integrators: A mixed implicit explicit formulation," in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2011, pp. 103–112.

[39] P. G. Sessa, D. Frick, T. A. Wood, and M. Kamgarpour, "From uncertainty data to robust policies for temporal logic planning," in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2018, pp. 157–166.

[40] K. Hashimoto and D. V. Dimarogonas, "Resource-aware networked control systems under temporal logic specifications," *Discrete Event Dyn. Syst.*, vol. 29, no. 4, pp. 473–499, 2019.

[41] M. Inoue and V. Gupta, "'Weak' control for human-in-the-loop systems," *IEEE Control Syst. Lett.*, vol. 3, no. 2, pp. 440–445, Apr. 2019.

[42] Y. Gao, F. J. Jiang, X. Ren, L. Xie, and K. H. Johansson, "Reachability-based human-in-the-loop control with uncertain specifications," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1880–1887, 2020.

[43] S. V. Rakovic, E. C. Kerrigan, D. Q. Mayne, and J. Lygeros, "Reachability analysis of discrete-time systems with disturbances," *IEEE Trans. Autom. Control*, vol. 51, no. 4, pp. 546–561, Apr. 2006.

[44] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. Eur. Control Conf.*, 2013, pp. 502–510.

[45] I. M. Mitchell and J. A. Templeton, "A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems," in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2005, pp. 480–494.

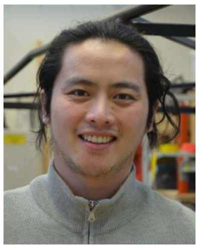
[46] E. A. Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for linear systems," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1163–1176, May 2014.

Yulong Gao received the B.E. degree in automation and the M.E. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2013 and 2016, respectively, and the joint Ph.D. degree in electrical engineering from the KTH Royal Institute of Technology, Stockholm, Sweden, and Nanyang Technological University, Singapore, in 2021.

He was a Visiting Student with the Department of Computer Science, University of Oxford, Oxford, U.K., in 2019. He is currently a Postdoctoral Researcher with KTH. His research interests include automatic verification, stochastic control, and model predictive control with application to safety-critical systems.

Alessandro Abate (Senior Member, IEEE) received the Laurea in electrical engineering from the University of Padova, Padova, Italy, in 2002 and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, Berkeley, CA, USA, in 2004 and 2007, respectively.

He is currently a Professor of Verification and Control with the Department of Computer Science, University of Oxford, Oxford, U.K., and a Fellow of the Alan Turing Institute for Data Sciences, London, U.K. He has been an International Fellow with the CS Lab, SRI International, Menlo Park, CA, USA, and a Postdoctoral Researcher with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA. From 2009 to 2013, he has been an Assistant Professor with the Delft Centre for Systems and Control, TU Delft, Delft, Netherlands.



Frank J. Jiang received the B.S. degree in electrical engineering and computer science from the University of California, Berkeley, Berkeley, CA, USA, in 2016, and the M.S. degree in systems, control and robotics in 2019 from the KTH Royal Institute of Technology, Stockholm, Sweden, where he is currently working toward the Ph.D. degree in electrical engineering with the Division of Decision and Control Systems.

His research interests are in formal verification, machine learning, and control, and their applications in robotics and intelligent transportation systems.

Mirco Giacobbe received the B.S. and M.S. degrees in computer science from the University of Trento, Trento, Italy, in 2010 and 2012, respectively, the second M.S. degre in software systems engineering from RWTH Aachen, Aachen, Germany, in 2012, and the Ph.D. degree in computer science from IST Austria, Klosterneuburg, Austria, in 2019.

He is a Lecturer with the School of Computer Science, University of Birmingham, Birmingham, U.K. Before that, he was a Postdoctoral Research Associate with the Department of Computer Science, University of Oxford, Oxford, U.K. He is interested in formal methods and machine learning for the analysis and the control of cyber–physical systems.

Lihua Xie (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, in 1992.

He is currenctly a Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, the Director of Delta-NTU Corporate Laboratory for Cyber-Physical Systems, Singapore, and the Director of Center for Advanced Robotics Technology Innovation, Singapore. He served as the Head of Division of Control and Instrumentation from 2011 to 2014. He held teaching appointments in the Department of Automatic Control, Nanjing University of Science and Technology, Nanjing, China, from 1986 to 1989. His research interests include robust control and estimation, networked control systems, multiagent networks, localization, and unmanned systems.

Dr Xie is an Editor-in-Chief for *Unmanned Systems* and has served as Editor for *IET Book Series in Control* and Associate Editor for a number of journals including IEEE TRANSACTIONS ON AUTOMATIC CONTROL, *Automatica*, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK CONTROL SYSTEMS, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II. He was an IEEE Distinguished Lecturer from 2012 to 2014. He is a Fellow of Academy of Engineering Singapore, IFAC, and CAA.

Karl Henrik Johansson (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1992 and 1997, respectively.

He is currently the Director of KTH Digital Futures, Stockholm, Sweden, and a Professor with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. He has held visiting positions with the University of California, Berkeley, Berkeley, CA, USA; California Institute of Technology, Pasadena, CA, USA; Nanyang Technological University, Singapore; HKUST Institute of Advanced Studies, Hong Kong; and Norwegian University of Science and Technology, Trondheim, Norway. His research interests include networked control systems, cyber–physical systems, and applications in transportation, energy, and automation.

Dr. Johansson has served on the IEEE Control Systems Society Board of Governors and the IFAC Executive Board and is currently the Vice-President of the European Control Association Council. He is the recipient of several best paper awards and other distinctions. He has been awarded Distinguished Professor with the Swedish Research Council and Wallenberg Scholar. He is a recipient of the Future Research Leader Award from the Swedish Foundation for Strategic Research and the triennial Young Author Prize from IFAC. He is a Fellow of the Royal Swedish Academy of Engineering Sciences. He is an IEEE Distinguished Lecturer.