Adaptive Task Planning and Formal Control Synthesis Using Temporal Logic Trees

Yulong Gao[1(✉)], Can Zhou[1], Alessandro Abate[2], and Karl H. Johansson[3]

[1] Department of Electrical and Electronic Engineering, Imperial College London, London, UK
{yulong.gao,c.zhou24}@imperial.ac.uk
[2] Department of Computer Science, University of Oxford, Oxford, UK
alessandro.abate@cs.ox.ac.uk
[3] Division of Decision and Control Systems, KTH Royal Institute of Technology, and Digital Futures, Stockholm, Sweden
kallej@kth.se

Abstract. Temporal logics have garnered significant attention in the control community due to their use for formal control synthesis, namely for the synthesis of control policies with provable correctness guarantees for more complex and interesting properties than traditional control objectives. Formal control under temporal logics is fundamentally challenging though, particularly when dealing with uncertain infinite systems and complex temporal logic specifications for real-time task planing, as the established methods struggle with handling models in high dimensions and with accommodating online deployment. In this article, we propose Temporal Logic Trees (TLT) as a mitigation for these challenges. TLT are constructed from Linear Temporal Logic (LTL) formulae via reachability analysis, offering an abstraction-free design method. Building upon the TLT framework, we present approaches for adaptive task planning and formal control synthesis that are usable on both finite and infinite systems. Furthermore, we demonstrate the applicability of our approach for online control synthesis, particularly in addressing time-varying tasks: namely, our method allows for dynamic online updates of the specifications, which showcases its practical utility and flexibility.

Keywords: Formal control synthesis · Task planning · Linear temporal logic · Temporal logic tree · Real-time systems

1 Introduction

In recent years, temporal logics – such as Linear Temporal Logic (LTL) [16], Computational Tree Logic (CTL) [3], and Signal Temporal Logic (STL) [13] – have captured significant attention within the control community. This interest is due to the capability of these logics to facilitate formal control synthesis for objectives that are richer than traditional control tasks like stability and set invariance, whilst emphasising provable correctness guarantees with the synthesised

S. Graf et al. (Eds.): Festschrift Wang Yi, LNCS 15230, pp. 64–78, 2025.
https://doi.org/10.1007/978-3-031-73751-0_7

policy. In particular, temporal logics provide a structured framework for expressing and reasoning about complex, temporally-extended properties of dynamical systems, encompassing intricate and interesting aspects, such as safety (nothing bad will ever happen), liveness (something good will eventually happen), and richer combinations of Boolean and temporal requirements [1]. The diverse applications of control synthesis under temporal logic specifications, including single-robot control in dynamic environments [15], multi-robot planning [8], and control of transportation networks [4], further underscore the utility of temporal logics in the area of control.

The prevalent approaches in formal control synthesis under temporal logic specifications for dynamical systems revolve around automata-based methods. These methods typically begin by abstracting a given dynamical process as a finite transition system, under formal guarantees. Subsequently, they convert a given (linear) temporal logic formula into an equivalent automaton. Then, the transition system is synchronised with the automata that is obtained to constructed a "product automaton". This automaton can be converted into a decision problem or, more broadly, a game [11]. By solving this game, a control strategy is derived [5]. However, significant challenges arise when dealing with dynamical systems affected by uncertainty, and inherent limitations also exist within the dominant automata-based methods when addressing complex scenarios.

Firstly, abstraction from infinite systems to finite systems runs into the curse of dimensionality: abstraction techniques typically involve partitioning the state space and constructing transitions through reachability analysis, however computational complexity grows exponentially as system dimensionality increases. Secondly, there is a shortage of solutions for handling general LTL formulae for dynamical systems under uncertainty, such as bounded disturbance or additive noise. Recent advances [9,14] typically only cover fragments of LTL, such as bounded LTL and co-safe LTL. However, when LTL formulae are defined over infinite trajectories, it is difficult to handle uncertainty propagation along such trajectories. Thirdly, current methods often hardly cope with online deployment: instead, in numerous applications, obtaining full a-priori knowledge of a specification is unfeasible. As such, offline automata-based design methods become severely constrained. Lastly, controllers derived from automata-based methods often result in a single control policy, whereas certain applications necessitate *a set* of feasible control inputs to provide more flexibility in actual implementation.

This article moves beyond the above limitations by closely investigating the connection between temporal logic and reachability analysis. We leverage a novel tool "Temporal Logic Trees" (TLT) proposed in [6] and extend the results for jointly adaptive task planning and formal control synthesis for discrete-time uncertain dynamical systems in partly known environments. The new framework systematically integrates a task planner and a formal controller. Different from [6] where the task is time-invariant, the task planner in this article can update a given LTL tasks by incorporating real-time environmental information and observing the system state. Meanwhile, a given LTL task can be transformed

to a corresponding TLT via reachability analysis. As shown in [6], the TLT represents an under-approximation of the LTL formula, which ensures the soundness of TLT-based control synthesis. We thus offer a new tool for applications in formal control synthesis for time-varying tasks within real-time systems, thus demonstrating how specifications can be dynamically updated online through our approach and controllers adaptively synthesised with guarantees.

2 Preliminaries and Problem Statement

This section presents preliminaries and problem statement. We first introduce the notion of controlled transition system and recall the concept of reachability analysis.

2.1 Controlled Transition System

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

- *a set* $\mathbb{S}$ *of states;*
- *a set* $\mathbb{U}$ *of control inputs;*
- *a transition relation* $\rightarrow \in \mathbb{S} \times \mathbb{U} \times \mathbb{S}$*;*
- *a set* $\mathbb{S}_0$ *of initial states;*
- *a set* $\mathcal{AP}$ *of atomic propositions;*
- *a labelling function* $L : \mathbb{S} \rightarrow 2^{\mathcal{AP}}$*.*

Definition 2. *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* $\mathsf{Post}(x, u) = \{x' \in \mathbb{S} \mid x \xrightarrow{u} x'\}$.

Note that the controlled transition system CTS presents nondeterminism which can be explicitly characterised by the post set $\mathsf{Post}(x, u)$: there may exist multiple successors $x' \in \mathbb{S}$ from x under a fixed u.

Definition 3. *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 4. *(Policy) For a controlled transition system* CTS*, a* policy $\boldsymbol{\mu} = u_0 u_1 \dots u_k \dots$ *is a sequence of maps* $u_k : \mathbb{S} \rightarrow \mathbb{U}$*. Denote by* $\mathcal{M}$ *the set of all policies.*

Definition 5. *(Trajectory) For a controlled transition system* CTS*, an infinite* trajectory $\boldsymbol{p}$ *starting from* x_0 *under a policy* $\boldsymbol{\mu} = u_0 u_1 \dots u_k \dots$ *is a sequence of states* $\boldsymbol{p} = x_0 x_1 \dots x_k \dots$ *such that* $\forall k \in \mathbb{N}$, $x_{k+1} \in \mathsf{Post}(x_k, u_k(x_k))$*. Denote by* $\mathsf{Trajs}(x_0, \boldsymbol{\mu})$ *the set of infinite trajectories starting from* x_0 *under* $\boldsymbol{\mu}$*.*

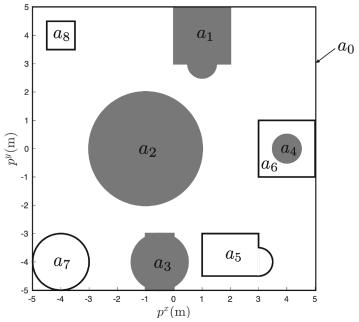


Fig. 1. Motion planning scenario

Example 1. (Motion Planning Example) We consider the motion planning problem for a mobile robot in an environment, as shown in Fig. 1. We describe the dynamics of the robot under disturbance as $x_{k+1} = x_k + f(x_k, u_k, w_k)\Delta$, where

$$f(x, u, w) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \sigma \end{bmatrix} + w,$$

and $x = [p^x, p^y, \theta]^T$ is the vehicle's x position, y position, and heading, respectively. The control input is $u = [v, \sigma]^T$, where v is the vehicle's velocity and σ is the angular velocity. Here, $\Delta = 0.05$s is the sampling period. The working space is $\mathbb{S} = \{z \in \mathbb{R}^3 \mid [-5, -5, -\pi]^T \leq z \leq [-5, 5, \pi]^T\}$, the control set is $\mathbb{U} = \{z \in \mathbb{R}^2 \mid [-0.5, -\pi/5]^T \leq z \leq [0.5, \pi/5,]^T\}$, and the disturbance set is $\mathbb{W} = \{z \in \mathbb{R}^3 \mid [-0.1, -0.1, -0.1]^T \leq z \leq [0.1, 0.1, 0.1]^T\}$. The atomic propositions are labels over the working space, and are denoted by a_i corresponding to the sets shown in Fig. 1. In particular, a_0 is the working space, a_i, $i = 1, 2, 3, 4$, represent obstacles, and a_i, $i = 5, 6, 7, 8$, represent target sets. Denote by $\mathbb{O}$ the union of the obstacles.

2.2 Reachability Analysis

Let us review the reachability analysis for a controlled transition system CTS.

Definition 6. *Consider a controlled transition system* CTS *and two sets* $\Omega_1, \Omega_2 \subseteq \mathbb{S}$. *The* k*-step reachable set from* Ω_1 *to* Ω_2 *is defined as*

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

The reachable set from Ω_1 *to* Ω_2 *is defined as*

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

Definition 7. *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* $\mathsf{Post}(x,u) \subseteq \Omega_f$. *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 $\Omega_f \subseteq \mathcal{RCI}(\Omega)$.*

Many algorithms have been proposed to compute the reachable set $\mathcal{R}(\Omega_1, \Omega_2)$ and the largest RCIS $\mathcal{RCI}(\Omega)$, see [2,12]. As we shall see, reachable sets and RCISs provide us a way to transfer the LTL formulae to the TLT framework, and to synthesize formal controllers. In the following, we treat the maps $\mathcal{R} : 2^{\mathbb{S}} \times 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ and $\mathcal{RCI} : 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ as the reachability operators.

2.3 LTL

An LTL formula is defined over a finite set of atomic propositions $\mathcal{AP}$ (in particular those relevant to a given dynamical system), and depends on propositional and temporal operators. The syntax of LTL in weak-until positive normal form is:

$$\varphi ::= \text{true} \mid \text{false} \mid a \mid \neg a \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1 \mathsf{W} \varphi_2. \quad (1)$$

Here $\bigcirc$, U, and W denote the "next", "until", and "weak-until" operators, respectively. Recall that $\varphi_1 \mathsf{W} \varphi_2 = \varphi_1 \mathsf{U} \varphi_2 \vee \Box\varphi_1$, where $\Box$ is the always operator. By employing the until operator, we can define its dual, known as "eventually," as: $\Diamond\varphi = \text{true} \cup \varphi$.

Definition 8. *(LTL semantics) For an LTL formula φ and a trajectory $\boldsymbol{p}$, the satisfaction relation $\boldsymbol{p} \vDash \varphi$ is defined as*

$$\boldsymbol{p} \vDash p \Leftrightarrow p \in L(x_0);\ \boldsymbol{p} \vDash \neg p \Leftrightarrow p \notin L(x_0),$$
$$\boldsymbol{p} \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \wedge \boldsymbol{p} \vDash \varphi_2;\ \boldsymbol{p} \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow \boldsymbol{p} \vDash \varphi_1 \vee \boldsymbol{p} \vDash \varphi_2,$$
$$\boldsymbol{p} \vDash \bigcirc\varphi \Leftrightarrow \boldsymbol{p}[1..] \vDash \varphi;\ \boldsymbol{p} \vDash \varphi_1 \mathsf{U} \varphi_2 \Leftrightarrow \exists j \in \mathbb{N}\ s.t. \begin{cases} \boldsymbol{p}[j..] \vDash \varphi_2, \\ \forall i \in \mathbb{N}_{[0,j-1]}, \boldsymbol{p}[i..] \vDash \varphi_1, \end{cases}$$
$$\boldsymbol{p} \vDash \Diamond\varphi \Leftrightarrow \exists j \in \mathbb{N},\ s.t.\ \boldsymbol{p}[j..] \vDash \varphi;\ \boldsymbol{p} \vDash \Box\varphi \Leftrightarrow \forall j \in \mathbb{N},\ s.t.\ \boldsymbol{p}[j..] \vDash \varphi,$$
$$\boldsymbol{p} \vDash \varphi_1 \mathsf{W} \varphi_2 \Leftrightarrow \begin{cases} \forall j \in \mathbb{N}, \boldsymbol{p}[j..] \vDash \varphi_1,\ or \\ \exists j \in \mathbb{N}\ s.t. \begin{cases} \boldsymbol{p}[j..] \vDash \varphi_2, \\ \forall i \in \mathbb{N}_{[0,j-1]}, \boldsymbol{p}[i..] \vDash \varphi_1. \end{cases} \end{cases}$$

Example 2. Let us continue considering Example 1. The motion planning task of interest is expressed by $\varphi = \phi_1 \wedge (\phi_2 \vee \phi_3) \wedge \phi_4 \wedge \phi_5$, where

$$\phi_1 = \Box\left(a_0 \wedge \neg\left(a_1 \vee a_2 \vee a_3 \vee a_4\right)\right), \phi_2 = \left(a_0 \wedge \neg a_5\right) \mathsf{U} a_6,$$
$$\phi_3 = \left(a_0 \wedge \neg a_5\right) \mathsf{U} a_7, \varphi_4 = \Diamond a_5, \phi_5 = \Diamond\Box a_8.$$

The formula φ specifies that the mobile robot is expected to first visit the region a_5 or a_6, then visit a_7, and finally stay in a_8 upon visiting it, while always avoiding the obstacles a_i, $i = 1, 2, 3, 4$.

2.4 Temporal Logic Trees

A TLT is a tree that simulates a hierarchical structure with set of linked nodes and of operator nodes. TLTs are proposed in [6] as an alternative tool for model checking and control synthesis over dynamical systems. The intuition underpinning a TLT is that it collects a sequence of sets of states (from its root node to the leaf nodes), indicating how a state trajectory of the model ought to evolve in order to satisfy the corresponding temporal logic specification.

Definition 9. *A TLT is a tree, for which the next properties hold:*

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

Definition 10. *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.*

Theorem 1 ([6]). *For a controlled transition system* CTS *and given any LTL formula φ, the following holds:*

(i) a TLT can be constructed from the formula φ through the reachability operators $\mathcal{R}$ and $\mathcal{RCI}$;

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

Example 3. Let us continue to consider Example 2 and detail how to perform reachability analysis for constructing a TLT from the given LTL formula $\varphi = \phi_1 \wedge (\phi_2 \vee \phi_3) \wedge \phi_4 \wedge \phi_5$, cf. Fig. 2(a). We begin with $\phi_1 = \Box\,(a_0 \wedge \neg\,(a_1 \vee a_2 \vee a_3 \vee a_4))$. Note that the set $\mathbb{S} \setminus \mathbb{O}$ corresponds to $a_0 \wedge \neg\,(a_1 \vee a_2 \vee a_3 \vee a_4)$. Since the formula ϕ_1 is a safety property, we use the robust invariance operator to compute $\mathbb{Y}_1 = \mathcal{RCI}(\mathbb{S} \setminus \mathbb{O})$. Then, the fragment $\mathbb{Y}_1 \Box (\mathbb{S} \setminus \mathbb{O})$ in Fig. 2(a). corresponds to the subformula ϕ_1. For $\phi_2 = (a_0 \wedge \neg a_5)\,\mathsf{U} a_6$ and $\phi_3 = (a_0 \wedge \neg a_5)\,\mathsf{U} a_7$, we define $\mathbb{Y}_2 = \mathcal{R}(\mathbb{S} \setminus L^{-1}(a_5), L^{-1}(a_6))$ and $\mathbb{Y}_3 = \mathcal{R}(\mathbb{S} \setminus L^{-1}(a_5), L^{-1}(a_7))$. Then, the fragments $\mathbb{Y}_2 \mathsf{U} L^{-1}(a_6)$ and $\mathbb{Y}_3 \mathsf{U} L^{-1}(a_7)$ in Fig. 2(a) correspond to the subformulae ϕ_2 and ϕ_3, respectively and they are connected by the disjunction operator. For $\varphi_4 = \Diamond a_5$, let $\mathbb{Y}_4 = \mathcal{R}(\mathbb{S}, L^{-1}(a_5))$. Then, the fragment $\mathbb{Y}_4 \mathsf{U} L^{-1}(a_5)$ in Fig. 2(a) corresponds to the subformula ϕ_4. Finally, we consider $\phi_5 = \Diamond \Box a_8$, which is a safety property. Let $\mathbb{Y}_6 = \mathcal{RCI}(L^{-1}(a_8))$ and $\mathbb{Y}_5 = \mathcal{R}(\mathbb{S}, \mathbb{Y}_6)$. Then, the fragment $\mathbb{Y}_5 \mathsf{U} \mathbb{Y}_6 \Box L^{-1}(a_8)$ in Fig. 2(a) corresponds to the subformula ϕ_5. According to the logical connection among these formulae, we connect these fragment and finally construct the TLT, as shown in Fig. 2(a).

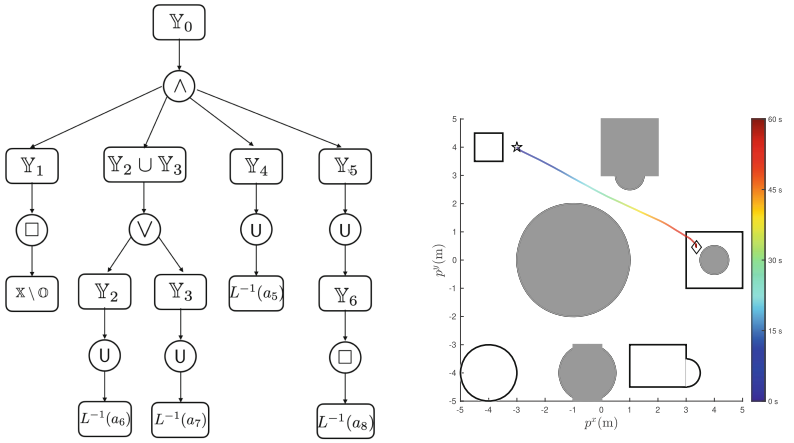


Fig. 2. (a) The controlled TLT for the LTL formula $\varphi = \phi_1 \wedge (\phi_2 \vee \phi_3) \wedge \phi_4 \wedge \phi_5$ in Example 3. (b) A state trajectory that fulfills the sub-formula ϕ_2 in Example 4.

2.5 Problem Statement

In this work, we consider the following task planing and control synthesis problem for a dynamical system.

Problem 1. Consider a controlled transition system CTS. At each time step k, (1) plan a time-varying LTL task φ_k and (2) synthesise a formal controller $\boldsymbol{\mu}_k$ such that φ_k is fulfilled.

We remark that in Problem 1, we focus on the dynamical system CTS which is fully observable, not partially observable. In comparison with a more standard control problem under a fixed LTL formula, the studied time-varying LTL planning provides much more flexibility. Firstly, it allows to consider time-varying (e.g., partially observable) environments: that is, the task φ_k can be updated from new features of the surrounding environment. For instance, when new obstacles are present (or equivalently detected in our fully observable setup), the LTL formula φ_k can naturally incorporate such new safety or collision-avoidance requirement. In addition, the formula φ_k can also be updated under the supervision of a human operator: for example, in a shared control scenario, a human operators is authorised to plan around the task and make decisions. Last but not least, our task planning allows for online deployment, hence enabling to monitor task completion and to decompose complex tasks into simple and easily solvable tasks in real time.

3 Adaptive Task Planner and Formal Control Synthesis

In order to solve the Problem 1, in this section, we propose a new TLT-based framework, as displayed in Fig. 3. The task planner adaptively updates the LTL

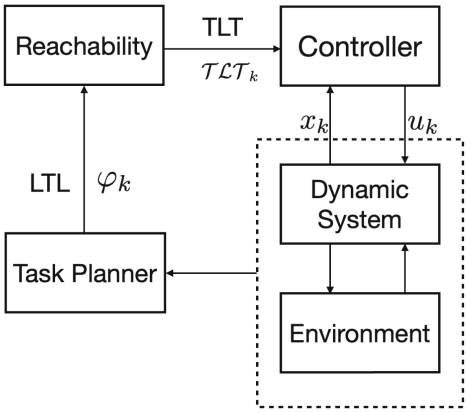


Fig. 3. A framework that integrates adaptive task planner and formal controller.

formula φ_k based on system information and environment knowledge. The task φ_k is then transformed into the corresponding TLT, denoted by $\mathcal{TLT}_k$, via reachability analysis (as presented above). Note that the TLT construction can be performed by pruning/modifying a previous TLT, which can be more efficient. An formal controller is then automatically synthesised using $\mathcal{TLT}_k$ and deployed online. In the following, we will detail how the functional blocks in Fig. 3 work and how they are systematically integrated.

3.1 Adaptive Task Planner

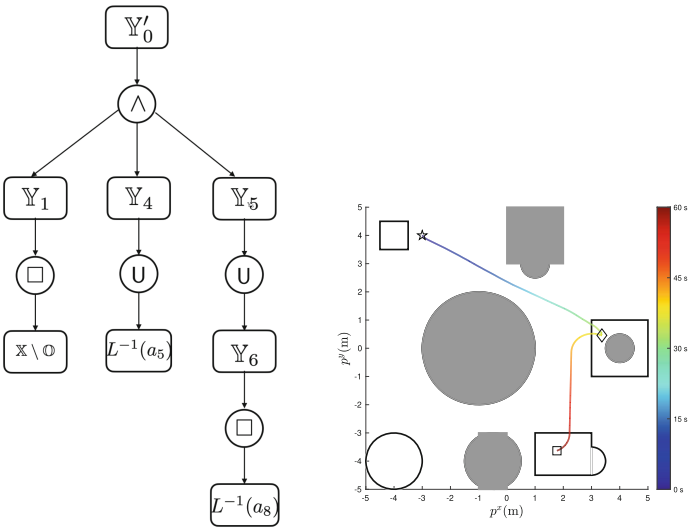
In general, the task planner is a function of the environment information and the system state, and it incrementally updates a given task. That is, at each time $k+1$, the new task φ_{k+1} can be written as a function g,

$$\varphi_{k+1} = g(\text{Env}_k, \varphi_k, x_k)$$

where Env_k is the environment state, φ_k is the previous task, and x_k is the system state. The function g depends the context, for example:

- The function g can serve as a task monitor. When a sub-formula ϕ_k of the task φ_k is fulfilled at time x_k, then such a sub-formula can be removed to obtain a simplified task φ_{k+1}. Please see Example 4 for more information.
- The function g can be part of the environment exploration: the planner is only aware of partial environment information at the initial stage. Real-time perception/observations enable to learn the environment and to update the task φ_k by adding new sub-tasks. For example, a new safety requirement should be incorporated when unknown obstacles are detected. Please see Sect. 4 for more details.
- The function g can also depend on the supervision from human operators. The task can be accordingly updated when the human operator intervenes

and provides new requirements. This may occur in remote control scenarios. Please refer to [7,10] for an example on remote car parking.



(a) (b)

Fig. 4. (a) The TLT for the simplified LTL formula $\varphi_k = \phi_1 \wedge \phi_4 \wedge \phi_5$. (b) The state trajectory that fulfills the subformula $\phi_2 \wedge \phi_4$.

3.2 TLT-based Formal Control Synthesis

Given the current state x_k, the control synthesis algorithm is structured as follows: (1) construct the controlled TLT by replacing each state-set node in the TLT $\mathcal{TLT}_k$ with a corresponding control set: if x_k belongs to a state-set node in the TLT, collect all the control inputs such that the next states is feasible under the connected operators; (2) compress this controlled TLT by taking union of the control sets over the fragments without logic operators: the compressed tree only has the control-set nodes and logic operator nodes; (3) backtrack a single control set $\mathbb{U}(x_k) \subseteq \mathbb{U}$ by taking union and intersection through a bottom-up traversal. Note, this control set $\mathbb{U}(x_k)$ collects all the feasible control inputs, such that the LTL specification φ_k is recursively feasible. That is, the controller is able to utilize the $\mathcal{TLT}_k$ to synthesize least-restrictive control sets that ensures the recursive feasibility and task satisfiability from x_k for each φ_k. In addition, we remark that the control set $\mathbb{U}(x_k)$ is least-restrictive in the sense that it permits any control inputs that do not violate φ_k. This control set thus provides more freedom to select the control input $u_k \in \mathbb{U}(x_k)$ to be deployed. For more details, we refer the readers to [6].

Example 4. Let us continue to consider Example 3. Let the initial task $\varphi_0 = \phi_1 \wedge (\phi_2 \vee \phi_3) \wedge \phi_4 \wedge \phi_5$. Before reaching the regions associated with a_6 or a_7 (that is, the subformlae ϕ_2 or ϕ_3 are fulfilled), the task φ_k will be the same as φ_0. Using the TLT shown in Fig. 2(a), the synthesised control set is $\mathbb{U}(x_k) = \mathbb{U}_1(x_k) \cap (\mathbb{U}_2(x_k) \cup \mathbb{U}_3(x_k))$, where $\mathbb{U}_1(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_1\}$, $\mathbb{U}_2(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_2\}$, and $\mathbb{U}_3(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_3\}$. The controller can freely select any $u_k \in \mathbb{U}(x_k)$ for implementation. In this example, we select the control input $u_k \in \mathbb{U}(x_k)$ in a time-optimal manner. That is, we minimise the task completion time of ϕ_2 or ϕ_3 subject to $u_k \in \mathbb{U}(x_k)$. A resulting state trajectory is shown in Fig. 2(b), which fulfills the sub-formula ϕ_2.

To this end, the task planner can remove the sub-formulae ϕ_2 and ϕ_3, and obtain a simplified formula $\varphi_k = \phi_1 \wedge \phi_4 \wedge \phi_5$. The corresponding TLT $\mathcal{TLT}_k$ can be easily obtained by pruning the TLT in Fig. 2(a), that is, removing two complete paths of the TLT associated with ϕ_2 and ϕ_3. The new TLT is shown in Fig. 4(a). The corresponding TLT-based control set is $\mathbb{U}(x_k) = \mathbb{U}_1(x_k) \cap \mathbb{U}_4(x_k)$, where $\mathbb{U}_4(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_4\}$. An extended state trajectory is shown in Fig. 4(b), which fulfills the sub-formula $\phi_2 \wedge \phi_4$.

Now the task planner can further remove the sub-formula ϕ_4 and obtain a simplified formula $\varphi_k = \phi_1 \wedge \phi_5$. The corresponding TLT $\mathcal{TLT}_k$ can be easily obtained by removing the complete path associated with ϕ_4 in the TLT of Fig. 4(a). The new TLT is shown in Fig. 5(a). The corresponding TLT-based control set is $\mathbb{U}(x_k) = \mathbb{U}_1(x_k) \cap \mathbb{U}_5(x_k)$, where $\mathbb{U}_5(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_5\}$. An extended state trajectory is shown in Fig. 5(b), which fulfills the subformula $\phi_2 \wedge \phi_4 \wedge \Diamond \mathbb{Y}_6$.

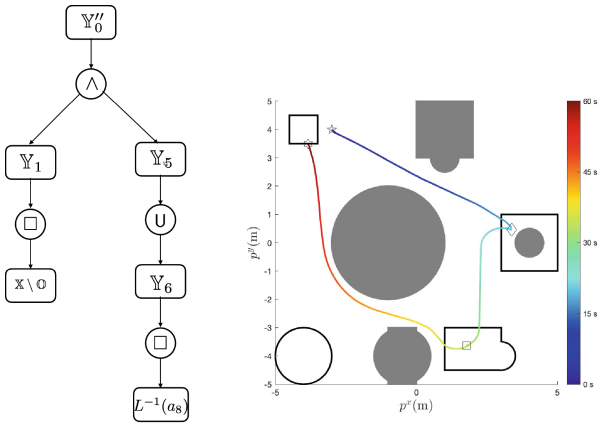
As long as the state enters the set $\mathbb{Y}_6$, the task planner can update the task to be $\varphi_k = \Box a_8$, whose TLT is shown in Fig. 6(a). The control set is $\mathbb{U}(x_k) = \{u \in \mathbb{U} \mid \mathsf{Post}(x_k, u) \subseteq \mathbb{Y}_6\}$, from which any control input u_k can keep the state in the region a_8. A resulting state trajectory satisfying φ_0 is shown in Fig. 6(b).

4 Application: Planning in Partially Known Environment

In this section, we provide another case study where the environment is only partially known and the task is time-varying. As shown in Fig. 7, we consider a scenario where an automated vehicle plans to move to a target set $\mathbb{T}$ but with some obstacles on the road that ought to be avoided. We assume that the sensing region of the vehicle is limited, hence the obstacles are only found on the way to the target. We use a single integrator model with a sample period of 1 second to describe the dynamics of the vehicle:

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_k + w_k. \tag{3}$$

The working space is $\mathbb{X} = \{z \in \mathbb{R}^2 \mid [0, -5]^T \leq z \leq [150, 5]^T\}$, the control 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 $\mathbb{W} = \{z \in$

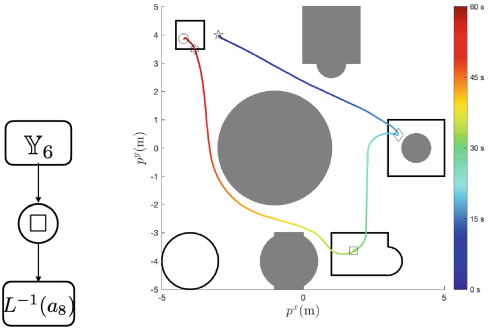


(a) (b)

Fig. 5. (a) The TLT for the simplified LTL formula $\varphi_k = \phi_1 \wedge \phi_5$. (b) The state trajectory that fulfills the subformula $\phi_2 \wedge \phi_4 \wedge \Diamond \mathbb{Y}_6$.

$\mathbb{R}^2 \mid [-0.1, -0.1]^T \leq z \leq [0.1, 0.1]^T\}$, 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 in the right direction unless a lane change is necessary to avoid broken vehicles. In Fig. 7, the two broken vehicles are encompassed by the sets $\mathbb{O}_1 = \{z \in \mathbb{R}^2 \mid [40, -5]^T \leq z \leq [45, 0]^T\}$ and $\mathbb{O}_2 = \{z \in \mathbb{R}^2 \mid [100, 0]^T \leq z \leq [105, 5]^T\}$. We assume that $\mathbb{O}_1$ and $\mathbb{O}_2$ are unknown to the vehicle at the beginning, but as soon as the vehicle can sense them, they become known to it.

Let the initial state be $x_0 = [0.5, -2.5]^T$ and the sensing range is 15 [m]. At time step $k = 0$, the set of atomic propositions is $\mathcal{AP} = \{a_1, a_2\}$ and if $x \in \mathbb{X}$, we



(a) (b)

Fig. 6. (a) The TLT for the simplified LTL formula $\varphi_k = \square a_8$. (b) The state trajectory that fulfills the formula φ_0.

define the labelling function as follows: if $x \in \mathbb{X} \cap \mathbb{T}$, $L(x) = \{a_1, a_2\}$; otherwise, $L(x) = \{a_1\}$. 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. 8 and implementing the TLT-based control synthesis algorithm, we obtain one realization as shown in Fig. 9. We can see that the vehicle keeps moving straightforward until it senses the obstacle $\mathbb{O}_1$ at $[25.46 \ \ -2.53]^T$.

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 of atomic propositions $\mathcal{AP}$, which becomes $\mathcal{AP} = \{a_1, a_2, a_3\}$. If $x \in \mathbb{X}$, we update the labelling function as follow: if $x \in \mathbb{X} \cap \mathbb{T}$, $L(x) = \{a_1, a_2\}$; if $x \in \mathbb{X} \cap \mathbb{O}_1$, $L(x) = \{a_1, a_3\}$; otherwise, $L(x) = \{a_1\}$. To avoid $\mathbb{O}_1$, the task planner update the task φ to be $\varphi' = \varphi \wedge (\square \neg a_3)$. We can

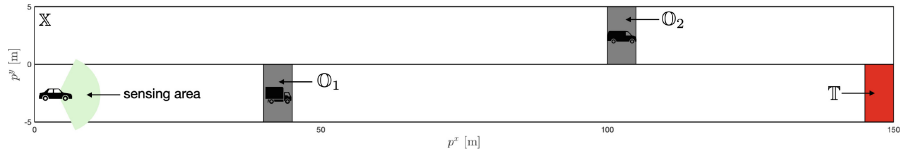


Fig. 7. Illustration of planning scenario for Example 2: an automated vehicle aims to reach a target set $\mathbb{T}$, however there are some unknown (broken) vehicles on the road to be avoided.

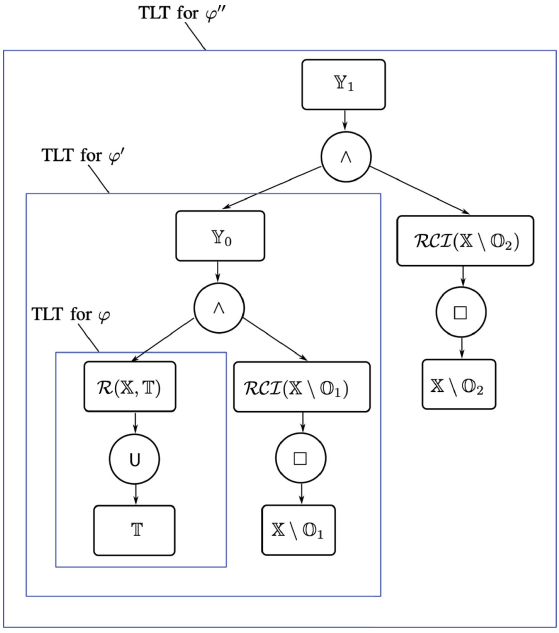


Fig. 8. The controlled TLT for the LTL formulae in Example 2, where $\varphi = a_1 \mathsf{U} a_2$, $\varphi' = \varphi \wedge (\square \neg a_3)$, $\varphi'' = \varphi' \wedge (\square \neg a_4)$, $\mathbb{Y}_0 = \mathcal{R}(\mathbb{X}, \mathbb{T}) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_1)$, and $\mathbb{Y}_1 = \mathcal{R}(\mathbb{X}, \mathbb{T}) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_1) \cap \mathcal{RCI}(\mathbb{X} \setminus \mathbb{O}_2)$.

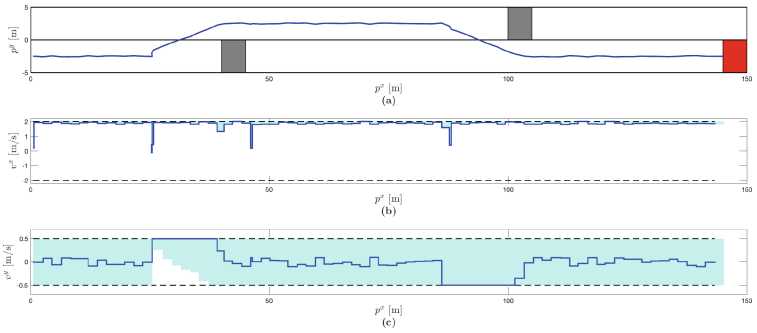


Fig. 9. Trajectories for one realization of disturbance trajectories: (a) state trajectories; (b) control trajectories of x-axis; (c) control trajectories of y-axis.

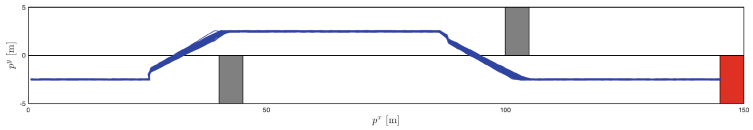


Fig. 10. State trajectories for 100 realizations of disturbance trajectories.

construct the TLT of φ' based on that of φ, which is shown in Fig. 8, and then continue to implement the TLT-based control synthesis algorithm. We can see that the vehicle changes lane from $[25.5, -2.4]$ and keeps moving forward. The trajectories are shown in Fig. 9. The vehicle is under the control obtained under φ' until it can sense $\mathbb{O}_2$ at $[86.11\ 2.58]^T$.

Similarly, when the vehicle can sense $\mathbb{O}_2$, we update the set $\mathcal{AP} = \{a_1, a_2, a_3, a_4\}$ and the labelling function as follows: if $x \in \mathbb{X} \cap \mathbb{T}$, $L(x) = \{a_1, a_2\}$; if $x \in \mathbb{X} \cap \mathbb{O}_1$, $g(x) = \{a_1, a_3\}$; if $x \in \mathbb{X} \cap \mathbb{O}_2$, $L(x) = \{a_1, a_4\}$; otherwise, $L(x) = \{a_1\}$. To avoid $\mathbb{O}_2$, the task planner update the task φ' to be $\varphi'' = \varphi' \wedge (\square \neg a_4)$. We can construct the TLT of φ'' based on that of φ', which is shown in Fig. 8. Under the TLT-based control, we can see that the vehicle merges back from $[86.11\ 2.58]^T$. Under the control with respect to φ'', the vehicle finally reaches the target set $\mathbb{T}$. Figure 9 (a) shows the state trajectories, from which we can appreciate that the entire specification is attained. Figure 9 (b)–(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. We repeat the above process for 100 realizations of the disturbance trajectories: the state trajectories for such 100 realizations are shown in Fig. 10.

5 Conclusion

In this article, we have introduced a TLT-based solution for integrating adaptive task planning and formal control synthesis. Unlike automaton-based meth-

ods, the TLT is constructed from LTL via reachability analysis, offering an abstraction-free method for control synthesis. By leveraging the controlled TLTs, we have presented techniques for adaptive task planning and formal control synthesis that are suitable to both finite- and infinite models under LTL specifications. We have showcased the effectiveness of our approach in a real-time problem, particularly in managing time-varying tasks and adaptively synthesising controllers accordingly. Future endeavors will focus on extending TLTs to accommodate broader temporal specification frameworks, such as CTL, along with comprehensive experimental validation and applications of our approach.

Acknowledgements. This work is supported in part by Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant and Swedish Research Council Distinguished Professor Grant 2017-01078.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
2. Blanchini, F., Miani, S., et al.: Set-Theoretic Methods in Control, vol. 78. Springer, Cham (2008). https://doi.org/10.1007/978-3-319-17933-9
3. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Workshop on logic of programs, pp. 52–71. Springer, Cham (1981). https://doi.org/10.1007/bfb0025774
4. Coogan, S., Gol, E.A., Arcak, M., Belta, C.: Traffic network control from temporal logic specifications. IEEE Trans. Control Netw. Syst. **2**(3), 162–171 (2016)
5. Emerson, E.A.: Automata, tableaux, and temporal logics. In: Proceedings of Workshop on Logic of Programs, pp. 79–88 (1985)
6. Gao, Y., Abate, A., Jiang, F.J., Giacobbe, M., Xie, L., Johansson, K.H.: Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems. IEEE Trans. Autom. Control **67**(10), 5071–5086 (2022)
7. Gao, Y., Jiang, F.J., Ren, X., Xie, L., Johansson, K.H.: Reachability-based human-in-the-loop control with uncertain specifications. IFAC-Papers Online **53**(2), 1880–1887 (2020)
8. Guo, M., Tumová, J., Dimarogonas, D.V.: Communication-free multi-agent control under local temporal tasks and relative-distance constraints. IEEE Trans. Autom. Control **12**(61), 3948–3962 (2016)
9. Hashimoto, K., Dimarogonas, D.V.: Resource-aware networked control systems under temporal logic specifications. Discret. Event Dyn. Syst. **29**, 473 (2019)
10. Jiang, F.J., Gao, Y., Xie, L., Johansson, K.H.: Human-centered design for safe teleoperation of connected vehicles. IFAC-PapersOnLine **53**(5), 224–231 (2020)
11. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Am. Math. Soc. **141**, 1–35 (1969)
12. Rakovic, S.V., Kerrigan, E.C., Mayne, D.Q., Lygeros, J.: Reachability analysis of discrete-time systems with disturbances. IEEE Trans. Autom. Control **51**(4), 546–561 (2006)
13. Raman, V., Donzé, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., Seshia, S.A.: Model predictive control with signal temporal logic specifications. In: 53rd IEEE Conference on Decision and Control, pp. 81–87. IEEE (2014)

14. Sessa, P.G., Frick, D., Wood, T.A., Kamgarpour, M.: From uncertainty data to robust policies for temporal logic planning. In: Proceedings of ACM International Conference on Hybrid Systems: Computation and Control, pp. 157–166 (2018)
15. Ulusoy, A., Belta, C.: Receding horizon temporal logic control in dynamic environments. Int. J. Robot. Res. **12**(33), 1593–1607 (2014)
16. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency, pp. 238–266. Springer (1996)