

# TuLiP: A Software Toolbox for Receding Horizon Temporal Logic Planning

Tichakorn Wongpiromsarn, Ufuk Topcu, Necmiye Ozay, Huan Xu, and Richard M. Murray  
California Institute of Technology, Pasadena, CA  
{nok, utopcu, necmiye, mumu, murray}@cds.caltech.edu

## ABSTRACT

This paper describes TuLiP, a Python-based software toolbox for the synthesis of embedded control software that is provably correct with respect to an expressive subset of linear temporal logic (LTL) specifications. TuLiP combines routines for (1) finite state abstraction of control systems, (2) digital design synthesis from LTL specifications, and (3) receding horizon planning. The underlying digital design synthesis routine treats the environment as adversary; hence, the resulting controller is guaranteed to be correct for any admissible environment profile. TuLiP applies the receding horizon framework, allowing the synthesis problem to be broken into a set of smaller problems, and consequently alleviating the computational complexity of the synthesis procedure, while preserving the correctness guarantee.

## 1. INTRODUCTION

To achieve higher levels of autonomy, modern embedded control systems need to reason about complex, uncertain, spatio-temporal environments and make decisions that enable complex missions to be accomplished safely and efficiently. To this end, linear temporal logic (LTL) is widely used as a specification language to precisely define system correctness. The embedded control software needs to be able to integrate discrete and continuous decision-making and provide correctness guarantee with respect to a given specification. Furthermore, since the environment may be dynamic and unknown a priori, it is important that the controller ensures proper response to all the admissible environment profiles.

Embedded control software synthesis has recently attracted considerable attention. A common approach is to construct a finite transition system that serves as an abstract model of the physical system and synthesize a strategy, represented by a finite state automaton, satisfying the given temporal, logical properties based on this model. Software packages that rely on this abstraction-based approach include LTL-Con [2], Pessoa [4] and LTLMoP [3]. LTLCon handles affine control systems and arbitrary LTL specifications. Pessoa

admits a more general class of systems, including nonlinear and switched, but only a very limited class of LTL specifications. However, both of these tools do not take into account the adversarial nature of the environment. Hence, the controller is only provably correct with respect to an a priori known and fixed environment. In contrast, LTLMoP accounts for the adversarial nature of the environment but only considers fully actuated systems operating in the Euclidean plane. A sampling-based method has been proposed for  $\mu$ -calculus specifications [6]. Due to the nature of the abstractions, this approach currently does not provide the correctness guarantee for all the admissible environments.

This paper introduces TuLiP, a Python-based toolbox for embedded control software synthesis. Similar to LTLMoP, TuLiP models the environment as an adversary. However, this often leads to the state explosion problem since all the admissible environment profiles need to be taken into consideration in the synthesis process. The novelty of TuLiP is to integrate a receding horizon framework [10] to alleviate the computational complexity of synthesis. Another extension is that TuLiP admits general affine control systems with bounded disturbances.

## 2. TuLiP FEATURES

The current version of TuLiP can be freely downloaded from <http://www.cds.caltech.edu/tulip>. We now summarize its two key features.

### 2.1 Embedded Control Software Synthesis

TuLiP deals with systems that comprise the plant, i.e., the physical component regulated by the controller, and the potentially dynamic and unknown a priori environment in which the plant operates. Note that the environment does not only include the factors that are external to the plant but it also includes the factors over which the system does not have control, e.g., hardware failure. The plant may contain both continuous (e.g. physical) and discrete (e.g. computational) components. TuLiP models the embedded control software synthesis problem as a game between the plant and the environment. Given the model of the plant and system specification  $\varphi$  in LTL, TuLiP provides a function that automatically synthesizes a controller that ensures system correctness with respect to  $\varphi$  for any admissible environment, if such a controller exists (i.e.,  $\varphi$  is realizable). If  $\varphi$  is unrealizable, TuLiP also provides counter examples, i.e., initial states starting from which the environment can falsify  $\varphi$  regardless of controller's actions.

The embedded control software synthesis feature relies on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HSCC '11 Chicago, Illinois USA

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

(1) generating a proposition preserving partition of the continuous state space, (2) continuous state space discretization based on the evolution of the continuous state, and (3) digital design synthesis. The algorithm for the continuous state space discretization is explained in [9]. JTLV [1] is used as the underlying digital design synthesis routine.

Currently, TuLiP handles the case where the continuous state of the plant evolves according to discrete-time linear time-invariant dynamics: for  $t \in \{0, 1, 2, \dots\}$ ,

$$s[t+1] = As[t] + Bu[t] + Ed[t], u[t] \in \mathcal{U}, d[t] \in \mathcal{D}, s[0] \in \mathcal{S}$$

where  $\mathcal{S} \in \mathbb{R}^n$  is the continuous state space,  $\mathcal{U} \in \mathbb{R}^m$  is the set of admissible control inputs,  $\mathcal{D} \in \mathbb{R}^p$  is the set of exogenous disturbances and  $s[t], u[t], d[t]$  are the continuous state, the control signal and the exogenous disturbance, respectively, at time  $t$ .  $\mathcal{U}, \mathcal{D}, \mathcal{S}$  are assumed to be bounded polytopes.

The specification  $\varphi$  is assumed to be of the form

$$\varphi = (\psi_{init} \wedge \Box\psi_e \wedge \bigwedge_{i \in I_f} \Box\Diamond\psi_{f,i}) \implies (\Box\psi_s \wedge \bigwedge_{i \in I_g} \Box\Diamond\psi_{g,i}),$$

known as *General Reactivity*[1]. Here  $\psi_{init}, \psi_e, \psi_{f,i}, i \in I_f, \psi_s$  and  $\psi_{g,i}, i \in I_g$  are propositional formulas.  $\psi_{init}, \psi_e$  and  $\psi_{f,i}, i \in I_f$  essentially describe the assumptions on the initial state of the system and the environment whereas  $\psi_s$  and  $\psi_{g,i}, i \in I_g$  describe the desired behavior of the system. See [10] for more details.

## 2.2 Receding Horizon Framework

For systems with a certain structure, the computational complexity of the planner synthesis can be alleviated by solving the planning problems in a receding horizon fashion, i.e., compute the plan or strategy over a “shorter” horizon, starting from the current state, implement the initial portion of the plan, and recompute the plan. This approach essentially reduces the planner synthesis problem into a set of smaller problems. To ensure that this “receding horizon” strategy preserves the desired system-level properties, certain sufficient conditions need to be satisfied.

Given a specification in the form of  $\varphi$  above, TuLiP first constructs a finite state abstraction of the physical system. Then, for each  $i \in I_g$ , we organize the system states into a partially ordered set  $\mathcal{P}^i = (\{\mathcal{W}_j^i\}, \preceq_{\psi_{g,i}})$  where  $\mathcal{W}_0^i$  are the set of states satisfying  $\psi_{g,i}$ . For each  $j$ , we define a short-horizon specification  $\Psi_j^i$  associated with  $\mathcal{W}_j^i$  as

$$\begin{aligned} \Psi_j^i &= ((\nu \in \mathcal{W}_j^i) \wedge \Phi \wedge \Box\psi_e \wedge \bigwedge_{k \in I_f} \Box\Diamond\psi_{f,k}) \\ &\implies (\Box\psi_s \wedge \Box\Diamond(\nu \in \mathcal{F}^i(\mathcal{W}_j^i)) \wedge \Box\Phi). \end{aligned}$$

Here  $\Phi$  is a propositional formula that describes receding horizon invariants and  $\mathcal{F}^i : \{\mathcal{W}_j^i\} \rightarrow \{\mathcal{W}_j^i\}$  defines the intermediate goal for starting in  $\mathcal{W}_j^i$ . Let  $\mathcal{V}$  be the entire state space of the system. As described in [10], a sufficient condition for the receding horizon strategy to lead to correct execution with respect to  $\varphi$  is that for all  $i \in I_g$ , (1)  $\mathcal{W}_0^i \cup \mathcal{W}_1^i \cup \dots \cup \mathcal{W}_p^i = \mathcal{V}$ , (2)  $\mathcal{W}_0^i \prec_{\psi_{g,i}} \mathcal{W}_j^i, \forall j \neq 0$ , (3)  $\mathcal{F}^i(\mathcal{W}_j^i) \prec_{\psi_{g,i}} \mathcal{W}_j^i, \forall j \neq 0$ , (4)  $\psi_{init} \implies \Phi$  is a tautology, and (5)  $\Psi_j^i$  is realizable  $\forall j$ .

Given the model of the plant, the original specification  $\varphi$ ,  $\{\mathcal{W}_j^i\}$ ,  $\mathcal{F}^i$  and  $\Phi$ , TuLiP automatically constructs the short horizon specification  $\Psi_j^i$  for each  $i, j$ . It also provides functions for (1) verifying that there exists a partial order  $\preceq_{\psi_{g,i}}$  and that the sufficient condition above is satisfied and (2) automatically computing the receding horizon invariant  $\Phi$  if one exists or report an error otherwise.

## 3. APPLICATIONS AND DISCUSSIONS

We have demonstrated the successful applications of TuLiP in multiple applications, including autonomous driving [8], vehicle management systems in avionics [11] and multi-target tracking. Other simpler examples are included in the current release of the toolbox. For the autonomous driving problem, the receding horizon framework needs to be applied for the car to be able to drive a reasonable distance. Due to the state explosion problem, TuLiP cannot automatically find the receding horizon invariant  $\Phi$  for this specific application. Nevertheless, it provides useful guidelines for the user to easily manually construct  $\Phi$ . Once  $\Phi$  is constructed, TuLiP successfully checks that the sufficient condition for applying the receding horizon strategy is satisfied.

Currently, TuLiP constructs  $\Phi$  roughly by starting with  $\Phi = True$  and iterating between (1) checking the realizability of each  $\Psi_j^i$ , and (2) updating  $\Phi$  to be the conjunction of current  $\Phi$  and the negation of the counter examples of unrealizable  $\Psi_j^i$ . This process stops when  $\psi_{init} \implies \Phi$  is no longer a tautology or all the  $\Psi_j^i$  are realizable. Since the counter examples are given as the enumeration of all the infeasible initial states, the size of  $\Phi$  quickly increases, leading to the state explosion problem. An extension of the current version of TuLiP is to implement a procedure for reducing the counter examples into a small formula. We also plan to integrate various existing software packages into TuLiP including a user-friendly simulation environment such as Player/Stage [5] and a state space discretization procedure that admits a more general class of systems (e.g. one based on approximate simulations and bisimulations as discussed in [7] and implemented in [4]).

## 4. REFERENCES

- [1] JTLV. <http://jtlv.sourceforge.net>.
- [2] LTLCon. <http://iasi.bu.edu/~software/LTL-control.htm>.
- [3] LTLMoP. <http://code.google.com/p/ltlmp>.
- [4] Pessoa. <http://www.cyphylab.ee.ucla.edu/pessoa>.
- [5] Player/Stage. <http://playerstage.sourceforge.net>.
- [6] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic  $\mu$ -calculus specifications. In *Proc. of the Conf. on Decision and Control*, 2009.
- [7] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.
- [8] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *Proc. of the Conf. on Decision and Control*, 2009.
- [9] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Automatic synthesis of robust embedded control software. In *Proc. AAAI Spring Symposium on Embedded Reasoning: Intelligence in Embedded Systems*, pp. 104–111, 2010.
- [10] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In K. H. Johansson and W. Yi, editors, *HSCC*, pp. 101–110, ACM, 2010.
- [11] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Formal synthesis of embedded control software: Application to vehicle management systems. In *Proc. AIAA Infotech@Aerospace*, 2011. submitted.