Receding Horizon Temporal Logic Planning for Dynamical Systems

Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray

Abstract—This paper bridges the advances in computer science and control to allow automatic synthesis of complex dynamical systems which are guaranteed, by construction, to satisfy the desired properties even in the presence of adversary. The desired properties are expressed in the language of temporal logic. With its expressive power, a wider class of properties than safety and stability can be specified. The resulting system consists of a discrete planner which plans, in the abstracted discrete domain, a set of transitions of the system to ensure the correct behaviors and a continuous controller which continuously implements the plan. For a system with certain structure, we present an approach, based on a receding horizon scheme, to overcome computational difficulties in the synthesis of a discrete planner and allow more complex problems to be solved.

I. INTRODUCTION

Recent advances in computer science, such as the development of a polynomial-time algorithm to construct finite state automata from their temporal logic specifications, enable automatic synthesis of digital designs which satisfy a large class of properties including safety, guarantee and response even in the presence of adversary [1]. On the other hand, recent advances in control and abundance of computational resources enable automatic synthesis of continuous controllers which ensure the safety and stability of the systems even in the presence of disturbances and modeling errors [2], [3], [4]. In many applications, systems need to perform complex tasks and interact with (potentially adversarial) environments. These systems may contain both continuous and discrete components. A big challenge in this paradigm is to integrate the methods from the two communities such that automatic synthesis of such systems is possible.

Hybrid system theory has been developed to deal with systems with both discrete and continuous components. Control of hybrid systems has been studied extensively but properties of interest are typically limited to stability and safety [5], [6]. For the system to be able to perform complex tasks, a wider class of properties such as guarantee (e.g. eventually perform task 1 or task 2 or task 3) and response (e.g. if the system fails, then eventually perform task 1 or perform tasks 1, 2 and 3 infinitely often in any order) need to be considered. Temporal logics have therefore garnered great interest due to their expressive power. In particular, MILP formulation has been used to incorporate linear temporal logic in control [7]. The interaction with potentially adversarial environments, however, is not taken into consideration.

The development of language equivalence and bisimulation notions allows abstraction of the continuous component of the system to a purely discrete model while preserving all the desired properties [8]. This subsequently provides a hierarchical approach to system design. In particular, a two-layer design is common and widely used in the area of planning and control. In the first layer, a discrete planner plans, in the abstracted discrete domain, a set of transitions of the system to ensure the satisfaction of the desired properties. This abstract plan is then continuously implemented by a continuous controller in the second layer. Bisimulations provide the proof that the continuous execution preserves the desired properties. This two-layer approach has been applied to robot motion planning for a special case of fully actuated ($\dot{x} = u$) [9], [10] and kinematic ($\dot{x} = A(x)u$) [11] models. In [10], [11], the discrete planner is constructed using a digital design synthesis tool [1]. The main limitation of this tool is the state explosion problem which restricts its applications to small problems.

The contribution of this paper is twofold. First, we extend the work of [10], [11] to a more complicated system with linear and piecewise affine (PWA) dynamics. A notion of reachability is defined that is sufficient to ensure that the continuous execution preserves the correctness of the discrete plan. The requirement of a bisimulation abstraction is then relaxed to a simulation abstraction which is enforced by restricting the set of discrete plans to those satisfying the reachability relation which can be established by solving a multiparametric programming problem. The Multi-Parametric Toolbox [4] provides an off-the-shelf computational machinery that enables the multiparametric programming problem to be solved in an automated fashion. Together with the digital design synthesis tool [1], this allows automatic design of dynamical systems which satisfy a wide range of properties expressed in temporal logic, taken into account the interaction with potentially adversarial environments.

Second, to address the state explosion problem in the digital design synthesis, we present a receding horizon scheme for executing finite state automata while ensuring system correctness. This allows the synthesis to be performed on a smaller domain and thus potentially substantially reduce the size of the space of the synthesis problem. As a result, more complex problems can be handled.

II. PRELIMINARIES

We use linear temporal logic (LTL) to describe the desired properties of the system. Given an LTL formula, we want to construct a finite state automaton, which can be thought of as a graph with a finite number of nodes (representing the states of the system) and edges (representing the transitions between states), such that the state transitions in the automa-
ton ensure the correctness of the system. In this section, we briefly describe the definition of LTL and the synthesis of a finite state automaton which satisfies a given LTL formula.

A. Terminology and Notations

Definition 1: A system consists of a set of variables V. The domain of V, denoted by dom(V), is the set of valuations of V. A state of the system is an element \( v \in \text{dom}(V) \).

Definition 2: A finite transition system is a tuple \( \Sigma = (\mathcal{V}, \rightarrow) \) where \( \mathcal{V} \) is a finite set of states, and \( \rightarrow \subseteq \mathcal{V} \times \mathcal{V} \) is a transition relation. Given states \( v_i, v_j \in \mathcal{V} \), we write \( v_i \rightarrow v_j \) if there is a transition from \( v_i \) to \( v_j \).

Definition 3: An atomic proposition is a statement on system variables \( v \) which has a unique truth value (True or False) for a given value of \( v \). Let \( v \in \text{dom}(V) \) be a state of the system and \( \pi \) be an atomic proposition. We write \( v \models \pi \) if \( \pi \) is True at the state \( v \). Otherwise, we write \( v \not\models \pi \).

Definition 4: An execution of a discrete-time system is an infinite sequence of the states of a particular run, i.e., an execution \( \sigma \) can be written as \( \sigma = v_0v_1v_2 \ldots \) where for each \( t \geq 0 \), \( v_t \in \text{dom}(V) \) is the state of the system at time \( t \).

B. Linear Temporal Logic

The use of linear temporal logic (LTL) as a specification language was introduced by Pnueli [12], [13]. LTL is built up from a set of atomic propositions, the logic connectives (\( \neg, \wedge, \vee, \Rightarrow, \equiv \)), and the temporal modal operators (\( \Box, \diamond, \bigcirc, \mathcal{U} \)). An LTL formula is defined inductively as follows.

1) Any atomic proposition \( \pi \) is an LTL formula.
2) Given an LTL formula \( \varphi \) and \( \psi \), the following are also LTL formulas: \( \neg \varphi, \varphi \vee \psi, \Box \varphi \) and \( \varphi \mathcal{U} \psi \).

Other operators can be defined as follows: \( \varphi \wedge \psi = (\neg \neg \varphi \vee \neg \psi) \), \( \varphi \Rightarrow \psi = \neg \varphi \vee \psi \), \( \Box \varphi = \text{True} \mathcal{U} \varphi \), and \( \Box \neg \varphi = \neg \Box \varphi \).

A propositional formula is one that does not include temporal operators. Given a set of LTL formulas \( \varphi_1, \ldots, \varphi_n \), their boolean combination is an LTL formula formed by joining \( \varphi_1, \ldots, \varphi_n \) with logic connectives.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution \( \sigma = v_0v_1v_2 \ldots \) and an LTL formula \( \varphi \), we say that \( \varphi \) holds at position \( i \geq 0 \) of \( \sigma \), written \( v_i \models \varphi \), if and only if \( \varphi \) holds for the remainder of the execution \( \sigma \) starting at position \( i \). The semantics of the temporal logic formula is defined as follows:

1) For an atomic proposition \( \pi \), \( v_i \models \pi \) iff \( v_i \models \pi \)
2) \( v_i \models \neg \varphi \) iff \( v_i \not\models \varphi \)
3) \( v_i \models \varphi \land \psi \) iff \( v_i \models \varphi \) or \( v_i \models \psi \)
4) \( v_i \models \Box \varphi \) iff \( v_{i+1} \models \varphi \)
5) \( v_i \models \varphi \mathcal{U} \psi \) iff \( \exists j \geq i, v_j \models \psi \) and \( \forall k \in [i, j), v_k \models \varphi \)

Based on the above definition, \( \Box \varphi \) holds at position \( i \) iff \( \varphi \) holds at every position in \( \sigma \) starting at position \( i \), and \( \mathcal{U} \varphi \) holds at position \( i \) iff \( \varphi \) holds at some position \( j \geq i \) in \( \sigma \).

Definition 5: An execution \( \sigma = v_0v_1v_2 \ldots \) satisfies \( \varphi \), denoted by \( \sigma \models \varphi \), if \( v_0 \not\models \varphi \). Let \( \Sigma \) be the set of all executions of a system. The system is said to be correct with respect to its specification \( \varphi \), written \( \Sigma \models \varphi \), if all its executions satisfy \( \varphi \), that is,

\[
(\Sigma \models \varphi) \iff (\forall \sigma, (\sigma \in \Sigma) \implies (\sigma \models \varphi)).
\]

C. Synthesis of Finite State Automata

In many applications, systems need to interact with the environments and whether they satisfy the desired properties depends on what the environments do. In this section, we informally describe the work of Piterman, et al. on Synthesis of Reactive(1) Designs [1]. We refer the reader to [1] and references therein for the detailed discussion of automatic synthesis of a finite state automaton from its specification.

From Definition 5, for a system to be correct, its specification \( \varphi \) must be satisfied regardless of what the environment does. Thus, the environment can be treated as adversary and the synthesis problem can be viewed as a two-player game between the system and the environment: the environment attempts to falsify \( \varphi \) while the system attempts to satisfy \( \varphi \). We say that \( \varphi \) is realizable if the system can satisfy \( \varphi \) no matter what the environment does.

For a specification of the form

\[
(\bigwedge_{i \in I} \Box \varphi_i) \implies (\bigwedge_{j \in J} \Box \psi_j).
\]

known as Generalized Reactivity(1), Piterman et al. shows that checking its realizability and synthesizing the corresponding automaton can be performed in polynomial time. In particular, we are interested in a specification of the form

\[
\varphi = (\varphi_e \Rightarrow \varphi_s)
\]

where roughly speaking, \( \varphi_e \) is an LTL formula that characterizes the initial states of the system and the assumptions of the environment and \( \varphi_s \) is an LTL formula that describes the correct behavior of the system, including the valid transitions the system can make. We refer the reader to [1] for precise definitions of \( \varphi_e \) and \( \varphi_s \). Note that since \( \varphi_e \Rightarrow \varphi_s \) is satisfied whenever \( \varphi_e \) is False, if the assumptions of the environment or initial state of the system violate \( \varphi_e \), the correct behavior \( \varphi_s \) of the system is not ensured, even though the specification \( \varphi \) is satisfied.

If the specification is realizable, the synthesis algorithm generates a finite state automaton that represents a set of transitions the system should follow in order to satisfy \( \varphi \). Assuming that the environment and the initial state of the system satisfy \( \varphi_e \), at any instance of time, there exists a node in the automaton that represents the current state of the system, and the system can follow the transition from this node to the next based on the current knowledge about the environment. However, if \( \varphi_e \) is violated, the automaton is no longer valid, meaning that there may not exist a node in the automaton which represents the current state of the system, or even though such a node exists and the system follows the transitions in the automaton, the correct behavior \( \varphi_s \) is not guaranteed.

If the specification is not realizable, the synthesis algorithm provides an initial state of the system for which there
exists a set of moves of the environment such that the system cannot satisfy \( \varphi \). The knowledge of the realizability of the specification is useful since it provides information about under what kind of conditions the system will fail to satisfy its desired properties.

The main limitation of the synthesis of finite state automata is the state explosion problem. In the worst case, the resulting automaton may contain all the possible states of the system. For example, if the system has 10 variables, each can take any value in \( \{1, \ldots, 10\} \), then there may be as many as \( 10^{10} \) nodes in the automaton.

### III. PROBLEM FORMULATION

Consider a system \( S \) with a set of variables \( V = S \cup E \) where \( S \) and \( E \) represent, respectively, the set of variables controlled by the system and the set of variables controlled by the environment. The domain of \( V \) is therefore given by \( \text{dom}(V) = \text{dom}(S) \times \text{dom}(E) \) and a state of the system can be written as \( v = (s, e) \) where \( s \in \text{dom}(S) \) and \( e \in \text{dom}(E) \).

Throughout the paper, we call \( s \) the controlled state and \( e \) the environment state.

Assume that the controlled state evolves according to the following discrete-time piecewise-affine (PWA) dynamics:

\[
\begin{align*}
    s[t+1] &= A_k s[t] + B_k u[t] + C_k \text{ if } (s[t], u[t]) \in \Omega_k \\
    u[t] &\in U
\end{align*}
\]

where \( k \in \{1, \ldots, N_{\text{PWA}}\} \), \( N_{\text{PWA}} \) is the number of regions in the PWA partition, \( \Omega_1, \ldots, \Omega_{N_{\text{PWA}}} \) is a polyhedral partition of \( \text{dom}(S) \times U \), for any natural number \( t \), \( s[t] \in \text{dom}(S) \) is the controlled state at time \( t \), \( u \) is the control signal and \( U \) is the set of admissible control inputs.

Let \( \Pi \) be a finite set of atomic propositions of variables from \( V \) and \( \varphi \) be an LTL specification built from \( \Pi \). Assume that \( \varphi \) can be expressed without the next (\( \Diamond \)) operator. We are interested in designing a controller for the system which ensures that any execution of the system satisfies the specification \( \varphi \) while respecting the dynamics (4) is hard since both the adversarial nature of the environment and the dynamics of the system need to be taken into account. To separate the concern of the environment from the concern of the dynamics, we apply the hierarchical approach, similar to [9], [10], [11], to solve the problem defined in Section III. That is, we decompose the problem into (a) designing a discrete planner that computes a discrete plan satisfying the specification \( \varphi \) regardless of what the environment does and (b) designing a continuous controller that implements the discrete plan while ensuring that the evolution of the system satisfies the dynamics (4).

The discrete planner can be automatically synthesized using a digital design synthesis tool [1] as described in Section II-C. However, since the synthesis algorithm requires a finite domain, the system \( S \) must be abstracted to a finite transition system. To construct a finite transition system \( \mathbb{D} \) from \( S \), we first partition \( \text{dom}(S) \) and \( \text{dom}(E) \), as in [9], [10], [11], into a finite number of equivalence classes or cells \( S \) and \( E \), respectively, such that the partition is proposition preserving [8]. Roughly speaking, this means that for any atomic proposition \( \pi \in \Pi \) and any states \( v_1 \) and \( v_2 \) that belong to the same cell in the partition, if \( v_1 \) satisfies \( \pi \), then \( v_2 \) also satisfies \( \pi \). We denote the resulting discrete domain of the system by \( \mathcal{V} = S \times E \). Throughout the paper, we call \( v \in \text{dom}(\mathcal{V}) \) a continuous state and \( v \in \mathcal{V} \) a discrete state of the system. For a discrete state \( v \in \mathcal{V} \), we say that \( v \) satisfies an atomic proposition \( \pi \in \Pi \), denoted by \( v \models \pi \), if and only if there exists a continuous state \( v \) contained in the cell labeled by \( v \) such that \( v \models \pi \). (Due to the proposition preserving property of the partition, this implies that for any continuous state \( v \) contained in the cell labeled by \( v \), \( v \models \pi \).)

Given an infinite sequence of discrete states \( \sigma_d = v_0v_1v_2 \ldots \) and an LTL formula \( \varphi \) built from \( \Pi \), we say that \( \sigma_d \) satisfies \( \varphi \) holds at position \( i \geq 0 \) of \( \sigma_d \), written \( v_i \models \varphi \), if and only if \( \varphi \) holds for the remainder of \( \sigma_d \) starting at position \( i \). With these definitions, the semantics of LTL for a sequence of discrete states can be derived from the general semantics of LTL defined in Section II-B.

Next, we need to determine the transition relations \( \rightarrow \) of \( \mathbb{D} \). As opposed to the special case of fully actuated dynamics considered in [9], [10], in our case, the dynamics (4), in general, constrain the evolution of the continuous state of the system which subsequently constrains a set of valid transitions of \( \mathbb{D} \). As a result, the transition relations of \( \mathbb{D} \) cannot be simply defined based on the topological adjacencies of the cells as in [9], [10]. Furthermore, constructing a bisimulation partition as in [9], [10] for a general system with PWA dynamics is hard. In this section, we relax the requirement that the partition is bisimulation and define the notion of reachability that is sufficient (but not necessary) to guarantee that if a discrete controlled state \( S_1 \) is reachable from \( S_0 \), the transition from \( S_0 \) to \( S_1 \) can be continuously implemented or simulated by a continuous controller. (See, for example, [15] for the exact definition.) A computational scheme that provides sufficient condition for reachability between two discrete controlled states is also presented in Section IV-A.

### A. Reachability

Let \( \{S_1, S_2, \ldots, S_n\} \) be a set of discrete controlled states. We define a map \( T_s : \text{dom}(S) \rightarrow S \) that sends a continuous controlled state to a discrete controlled state of its equivalence class. That is, \( T_s^{-1}(S) \in \text{dom}(S) \) is a set of all the continuous controlled states contained in the cell labeled by \( S \), and \( \{T_s^{-1}(S_1), \ldots, T_s^{-1}(S_n)\} \) is the partition of \( \text{dom}(S) \). We define the reachability relation, denoted by \( \rightarrow \), as follows: a discrete state \( S_1 \) is reachable from a discrete state \( S_0 \), written \( S_0 \rightarrow S_1 \), only if starting from any point \( s[0] \in T_s^{-1}(S_0) \), there exists a control law \( u \in U \) that takes the system (4) to a point \( s[T] \in T_s^{-1}(S_1) \) satisfying the constraint

---

---

We restrict ourselves to PWA dynamics for computational reasons. Our framework applies to nonlinear dynamics but no computational scheme to date is able to handle them efficiently.

This assumption is sufficient to ensure that \( \varphi \) is stutter invariant. See, for example, [14] for more detail.
s[t] ∈ T^-(S_i) ∪ T^-(S_j), ∀t ∈ {0, . . . , T}. Note that this is stronger than the usual definition of reachability [16]. We write S_i ⇀ S_j if S_i is not reachable from S_j. Clearly, if T^-^s(S_i) ∩ T^-^s(S_j) = ∅ (i.e., T^-^s(S_i) and T^-^s(S_j) are not topologically adjacent), then S_i ⇀ S_j.

In general, for two discrete states S_i and S_j, verifying the reachability relation S_i ⇀ S_j is hard. Therefore, we resort to a heuristic based on the following optimal control problem: Given discrete controlled states S_i, S_j ∈ S, the set of admissible control inputs U, the matrices A_k and B_k as in (4), a horizon length N ≥ 0 and the cost matrices P_N, Q ≥ 0 and R > 0, solve

\[
\begin{align*}
\min_{s[0] \ldots s[N-1] \in S} & \quad \left[ P_N s[N] \right] \| 2 + \sum_{t=0}^{N-1} \left[ Q s[t] \right] \| 2 + \left[ R u[t] \right] \| 2 \\
\text{s.t.} & \quad s[N] \in T^-^s(S_i), \quad s[0] \in \text{dom}(S), \\
& \quad s[t+1] = A_k s[t] + B_k u[t] \quad \text{if} \quad (s[t], u[t]) \in \Omega_k, \\
& \quad u[t] \in U, \\
& \quad s[t] \in T^-^s(S_i) \cup T^-^s(S_j), \\
& \quad \forall t \in \{0, . . . , N-1\}. \tag{5}
\end{align*}
\]

Note that (5) is a finite horizon optimal control problem. Furthermore, one can consider the problem in (5) as a family of problems parametrized by s[0] and it can be regarded as a multiparametric programming problem [3]. For certain choices of T^-^s(S_i), T^-^s(S_j), and U (for example polytopic sets, i.e., sets defined by affine inequalities), the Multi-Parametric Toolbox [4] exploits the explicit solution for this multiparametric programming problem, i.e., computes a partition Π_i,j of some subset of T^-^s(S_i) ∪ T^-^s(S_j) such that for any s[0] ∈ Π_i,j, the problem in (5) is feasible. An example of a set Π_i,j along with S_i and S_j is shown in Figure 1.

![Figure 1](image1.png)

Fig. 1. An example of a set Π_i,j represented by the unshaded region. For any s[0] in the shaded region, the optimal control problem (5) is infeasible. Different unshaded regions have different associated controllers. For more detail, see [4].

### B. State Space Discretization

In general, given the previous partition of dom(S) and any i, j ∈ {1, . . . , n}, the reachability relation between S_i and S_j may not be established through the solution of the multiparametric programming problem (5) since T^-^s(S_i) is not necessarily covered by Π_i,j (due to the constraints on u and a specific choice of the finite horizon N). This section describes a state space discretization scheme based on the reachability relation defined earlier to increase the number of valid discrete state transitions of D. The underlying idea is that for each S_i and S_j, we determine Π_i,j such that for any s[0] ∈ Π_i,j, the problem in (5) is feasible. Then, we partition T^-^s(S_i) into T^-^s(S_i) ∩ Π_i,j, labeled by S_i,1, and T^-^s(S_i) \ Π_i,j, labeled by S_i,2, and obtain the following reachability relations: S_i,1 ⇀ S_j and S_i,2 ⇀ S_j.

### C. Correctness of the System

Let S' = {S'_1, S'_2, . . . , S'_m} be the set of all the discrete controlled states corresponding to the resulting partition of dom(S) after applying the discretization algorithm proposed in Section IV-B. Since the partition obtained from the proposed algorithm is a subpartition of {T^-^s(S_i), . . . , T^-^s(S_n)} and \( \mathcal{V} = \mathcal{S} \times \mathcal{E} \) is proposition preserving, it is trivial to show that \( \mathcal{V}' = \mathcal{S}' \times \mathcal{E} \) is also proposition preserving. We define the finite transition system \( \mathcal{D} \) which serves as the abstract model of \( \mathcal{S} \) as follows: \( \mathcal{V}' = \mathcal{S}' \times \mathcal{E} \) is the set of states of \( \mathcal{D} \) and for any two states \( \mathcal{V}_i = (S'_i, \mathcal{E}_i) \) and \( \mathcal{V}_j = (S'_j, \mathcal{E}_j) \), \( \mathcal{V}_i \rightarrow \mathcal{V}_j \) (i.e. there exists a transition from \( \mathcal{V}_i \) to \( \mathcal{V}_j \)) only if \( S'_i \rightarrow S'_j \). Using the abstract model \( \mathcal{D} \), a discrete planner that guarantees the satisfaction of \( \varphi \) while ensuring
that the discrete plans are restricted to those satisfying the reachability relations can be automatically synthesized using a digital design synthesis tool as described in Section II-C.

From the stutter invariant property of $\varphi$, the formulation of the optimal control problem (5) and the proposition preserving property of $\mathcal{G}$, it is straightforward to show the following proposition.

**Proposition 1:** Let $\sigma_d = v_0v_1 \ldots$ be an infinite sequence of discrete states of $\mathcal{D}$ where for each natural number $k$, $v_k \rightarrow v_{k+1}$, $v_k = (\rho_k, \epsilon_k)$, $\rho_k \in \mathcal{S}'$ is the discrete controlled state and $\epsilon_k \in \mathcal{E}$ is the discrete environment state. If $\sigma_d \models \varphi$, then by applying a sequence of control laws, each corresponding to the solution of (5) with $S_0 = \rho_k$ and $S_0 = \rho_{k+1}$, the infinite sequence of continuous states $\sigma = v_0v_1v_2 \ldots$ satisfies $\varphi$.

### V. RECEIVING HORIZON STRATEGY

As discussed in Section II-C, automatic synthesis of finite state automata from their LTL specifications [1] suffers from the state explosion problem. In many applications, however, it is not necessary to plan for the whole execution, taking into account all the possible behaviors of the environment since the state that is very far from the current state of the system typically does not affect the near future plan. For example, consider a robot motion planning problem where the robot has to travel 100 kilometers. Under certain conditions, it may be sufficient to only plan out an execution for 500 meters and implement it in a receding horizon fashion, i.e., re-compute the plan as the robot moves. In this section, we present a sufficient condition and a receding horizon scheme that allows the synthesis to be performed on a smaller domain; thus, substantially reduces the number of states (or nodes) of the automaton while still ensuring the system correctness.

We consider a subclass of **Generalized Reactivity(1)** formulas (2): (a) Let $\varphi_{init}$ be a propositional formula of variables from $V$ which characterizes the initial state of the system; (b) let $\varphi_c$ be a boolean combination of propositional formulas of variables from $V$ and expressions of the form $\Box \psi_s$ where $\psi_s$ is a propositional formula of variables from $E$ which describes the assumptions on the transitions of environment states; (c) for each $j \in J$ where $J$ is a finite set, let $\varphi_j$ be a propositional formula of variables from $V$ which describes the assumption that must be satisfied infinitely often; (d) let $\varphi_s$ be a boolean combination of propositional formulas of variables from $V$ and expressions of the form $\Box \psi_s$ where $\psi_s$ is a propositional formula of variables from $V$ which describes the constraints on the transitions of system states; and (e) let $\varphi$ be a propositional formula of variables from $V$. We consider a specification of the finite transition system of the form

$$\varphi_{init} \land \Box \varphi_c \land \bigwedge_{j \in J} \Box \varphi_j$$

which $\Box \varphi_c$ and $\Box \varphi_j$ express the safety and the progress properties of the system. Let $\mathcal{V} = \mathcal{S} \times \mathcal{E}$ be the discrete domain of the system after applying the discretization algorithm presented in Section IV. Similar to the map $T_{\epsilon}$ for the controlled states defined in Section IV, we let $T : \text{dom}(\mathcal{V}) \rightarrow \mathcal{V}$ be a map which sends a continuous state to a discrete state of its equivalence class, i.e. for each $\mathcal{V}_i \in \mathcal{V}$, $T^{-1}(\mathcal{V}_i) \in \text{dom}(\mathcal{V})$ is a set of all the continuous state contained in the cell labeled by $\mathcal{V}_i$ and $\{T^{-1}(\mathcal{V}_i) \mid \mathcal{V}_i \in \mathcal{V}\}$ is a partition of $\text{dom}(\mathcal{V})$.

For the progress property $\Diamond \varphi$, suppose there exists a collection of disjoint subsets $\mathcal{W}_1, \ldots, \mathcal{W}_p$ of $\mathcal{V}$ such that (a) $\mathcal{W}_1 \cup \mathcal{W}_2 \cup \ldots \cup \mathcal{W}_p = \mathcal{V}$, (b) $\varphi_c$ is satisfied for any $v \in \bigcup_{i \in \mathcal{W}_i} T^{-1}(\mathcal{V}_i)$, and (c) $\{\mathcal{V}_1, \ldots, \mathcal{W}_p\}$ is a partially ordered set where for any $i$ and $j$, $\mathcal{W}_i \leq_p \mathcal{W}_j$ if and only if any execution starting from $v \in \bigcup_{i \in \mathcal{W}_i} T^{-1}(\mathcal{V}_i)$ and satisfying the property $\Diamond \varphi$ contains a state $v \in \bigcup_{i \in \mathcal{W}_i} T^{-1}(\mathcal{V}_i)$. By the abuse of notation, for each $i \in \{1, \ldots, p\}$, we let $T^{-1}(\mathcal{W}_i) = \bigcup_{i \in \mathcal{W}_i} T^{-1}(\mathcal{V}_i)$.

Suppose there exists a propositional formula $\Phi$ of variables from $V$ and for each $i \in \{1, \ldots, p\}$, there exist $g_i \in \{1, \ldots, p\}$ and a subset $\mathcal{D}_i$ of $\text{dom}(\mathcal{V})$ satisfying the following conditions:

1. $\varphi_{init} \implies \Phi$ is a tautology, i.e., any state satisfies $\varphi_{init}$ also satisfies $\Phi$.
2. $T^{-1}(\mathcal{W}_i), T^{-1}(\mathcal{W}_g) \subseteq \mathcal{D}_i \times \text{dom}(E)$, and
3. $\mathcal{W}_i \leq_p \mathcal{W}_g$, and for each $i \neq p$, $\mathcal{W}_i \leq_p \mathcal{W}_g$ such that

$$\Psi_i = ((v \in T^{-1}(\mathcal{W}_i)) \land \Phi \land \Box \varphi_c \land \bigwedge_{j \in J} \Box \varphi_j) \implies (\Box \varphi_s \land \Box \varphi_g)$$

(7) is realizable with the domain of $\mathcal{S}$ restricted to $\mathcal{D}_i$.

For $i \in \{1, \ldots, p\}$, let $\mathcal{A}_i$ be an automaton that satisfies $\Psi_i$. Since in the synthesis of $\mathcal{A}_i$, the domain of $\mathcal{S}$ is restricted to $\mathcal{D}_i$, this can substantially reduce the number of states in the automaton, especially when the size of $\mathcal{D}_i$ is much smaller than the size of $\text{dom}(\mathcal{S})$.

**Receding Horizon Strategy:** Starting from the state $v_0$, pick an automaton $\mathcal{A}_i$ such that $v_0 \in T^{-1}(\mathcal{W}_i)$ and execute $\mathcal{A}_i$ until the system reaches the state $v \in T^{-1}(\mathcal{W}_i)$ where $\mathcal{W}_i \geq_p \mathcal{W}_i$, at which point, switch to the automaton $\mathcal{A}_j$. Keep iterating this process until $\mathcal{A}_p$ is executed.

**Theorem I:** Suppose for each $i \in \{1, \ldots, p\}$, $\Psi_i$ is realizable. Then the proposed receding horizon strategy ensures the correctness of the system.

**Proof:** Consider an arbitrary execution $\sigma$ of the system that satisfies the formula to the left of $\implies$ in (6). From the tautology of $\varphi_{init} \implies \Phi$, it is easy to show that if $\sigma$ starts from $v \in T^{-1}(\mathcal{W}_i)$, then $\sigma$ satisfies the formula to the left of $\implies$ in (7). Let $v_0 \in \text{dom}(\mathcal{V})$ be the initial state of the system. First, suppose $v_0 \in T^{-1}(\mathcal{W}_p)$. Then, the system always executes $\mathcal{A}_p$, thus, $\Psi_p$ ensures that $\sigma$ satisfies (6). Next, suppose $v_0 \in T^{-1}(\mathcal{W}_i)$ where $i \neq p$. Then, the system executes $\mathcal{A}_i$, and $\Psi_i$ ensures that the safety property $\varphi_s$ holds at every position of $\sigma$ up to and including position $p_i$ at which the system switches the automaton $\Phi$ holds at position $p_i$. In addition, since $\Psi_i$ satisfies the progress property $\Diamond (v \in T^{-1}(\mathcal{W}_g))$ where $\mathcal{W}_g \geq_p \mathcal{W}_i$, $\Psi_i$ ensures that eventually the system reaches the state $v_j \in T^{-1}(\mathcal{W}_j)$ where $\mathcal{W}_g \geq_p \mathcal{W}_j$. According to the receding horizon scheme, the system switch an automaton at this state, i.e., $v_j$ is the state of the system at position $p_i$ of $\sigma$. Since $v_j \in T^{-1}(\mathcal{W}_j)$ and $v_j$ satisfies $\Phi$, $\sigma$ satisfies the formula to the left of $\implies$ in (7). Using the previous
argument, we get that \( \Psi_j \) ensures that the safety property \( \varphi_s \) holds at every position of \( \sigma \) starting from position \( p_i \) up to and including position \( p_j \) at which the system switches the automaton and \( \Phi \) holds at position \( p_j \). Keep iterating this proof, we get that \( \varphi_s \) holds at every position of \( \sigma \) and due to the finiteness of the set \( \{W_1, \ldots, W_k\} \) and its partial order structure, eventually the automaton \( A_0 \) is executed which ensures that \( \sigma \) satisfies the progress property \( \Diamond \varphi_g \).

Remark 4: The propositional formula \( \Phi \) (which can be viewed as an invariant of the system) adds a constraint on the initial state of the system assumed by each of the automata so that \( \Psi \) is realizable. One way to determine \( \Phi \) is to start with \( \Phi = \text{True} \) and check the realizability of the resulting \( \Psi_1 \). If for any \( i \in \{1, \ldots, p\} \), \( \Psi_i \) is realizable, we are done. Otherwise, the synthesis process provides the initial state of the system for which there exists a set of moves of the environment such that the system cannot satisfy \( \Psi_i \). This information provides guidelines for constructing \( \Phi \).

VI. EXAMPLE

We consider a point-mass omnidirectional vehicle navigating a straight road while avoiding obstacles and obeying certain traffic laws. It was shown in [17] that the nondimensional equations of motion of the vehicle are given by

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ \frac{q_0}{2mL^2} \end{bmatrix},
\]

where \( q_x, q_y, q_0 \) are the control inputs. The following constraints on the control efforts:

\[\forall t, q_x^2(t) + q_y^2(t) \leq \frac{3 - |q_0(t)|}{2} \text{ and } |q_0(t)| \leq 3.\]

Conservatively, we can set \( |q_x(t)| \leq \sqrt{0.5}, |q_y(t)| \leq \sqrt{0.5} \) and \( |q_0(t)| \leq 1 \) so that the constraints (9) are decoupled.

In this section, we are only interested in the translational \((x, y)\) components of the vehicle state. Discretizing the dynamics (8) with time step 0.1, we obtain the following discrete-time linear time-invariant state space model

\[
\begin{bmatrix}
z[i+1] \\
v_2[i+1]
\end{bmatrix} = \begin{bmatrix} 1 & 0.0952 \\ 0 & 0.9484 \end{bmatrix} \begin{bmatrix} z[i] \\
v_2[i]\end{bmatrix} + \begin{bmatrix} 0.0484 \\ 0.0952 \end{bmatrix} q_z \quad (10)
\]

where \( z \) represents either \( x \) or \( y \) and \( v_z \) represents the rate of change in \( z \). Let \( C_z \) be the domain of the vehicle state projected onto the \((z, v_z)\) coordinates. We restrict the domain \( C_z \) to \([z_{\text{min}}, z_{\text{max}}] \times [-1, 1]\) and partition \( C_z \) as

\[
C_z = \bigcup_{i \in \{z_{\text{min}}, \ldots, z_{\text{max}}\}} C_{z, i} \quad (11)
\]

where \( C_{z, i} = [i - 1, i] \times [-1, 1] \) as shown in Figure 2. Throughout the section, we call this partition the original partition of the domain \( C_z \).

We consider the road with 2 lanes, each of width 1, so we set \( y_{\text{min}} = 0 \) and \( y_{\text{max}} = 2 \). Since the vehicle dynamics are translationally invariant, without loss of generality we set \( x_{\text{min}} = 0 \) and \( x_{\text{max}} = L \) where \( L \) is the length of the road.

For each \( i \in \{1, \ldots, L\} \) and \( j \in \{1, 2\} \), we define a Boolean variable \( O_{i,j} \) that is assigned the value \( \text{True} \) if and only if an obstacle is detected at some position \((x_o, y_o) \in [i-1, i] \times [j-1, j]\). The state of the system is therefore a tuple \((x, v_x, y, v_y, O_{1,1}, O_{1,2}, \ldots, O_{L,1}, O_{L,2}) \) where \((x, v_x, y, v_y) \in [0, L] \times [-1, 1] \times [0, 2] \times [-1, 1] \) is the vehicle state or the controlled state and \((O_{1,1}, O_{1,2}, \ldots, O_{L,2}) \in \{0, 1\}^{2L} \) is the environment state.

A. System Specification

We assume that at the initial configuration, the vehicle is at least \( d_{\text{obs}} \) away from any obstacle and that the vehicle starts in the right lane, with \( x \in C_{x,1} \). That is, \( \varphi_{\text{init}} \) in (6) is defined as: for any \( i \in \{1, \ldots, L\} \),

\[
x \in \bigcup_{j=i}^{i+d_{\text{obs}}} C_{x,k} \implies (-O_{i,1} \land -O_{i,2}) \land y \in C_{y,1} \quad (12)
\]

The following properties are assumed for the environment.

1) An obstacle is always detected before the vehicle gets too close to it. That is, there is a lower bound \( d_{\text{pop-up}} \) on the distance from the vehicle for which obstacle is allowed to instantly pop up. An LTL formula corresponding to this assumption is a conjunction of the following formula: for all \( i \in \{1, \ldots, L\} \) and \( k \in \{1, 2\} \),

\[
\square \left( x \in \bigcup_{j=i}^{i+d_{\text{pop-up}}} C_{x,j} \land -O_{i,k} \implies \Box(-O_{i,k}) \right) \quad (13)
\]

2) Sensing range is limited. That is, the vehicle cannot detect an obstacle that is away from it farther than \( d_{\text{sr}} \). An LTL formula corresponding to this assumption is a conjunction of the following formula: for all \( i \in \{1, \ldots, L\} \),

\[
\square \left( x \in C_{x,i} \implies \bigwedge_{j=i+d_{\text{sr}}} \Box(-O_{j,1} \land -O_{j,2}) \right) \quad (14)
\]

3) The road is not blocked. That is, for any \( i \in \{1, \ldots, L\} \),

\[
\square(-O_{i,1} \lor -O_{i,2}) \quad (15)
\]

4) To make sure that the stay-in-lane requirement (see below) is achievable, we assume that an obstacle on the right lane does not disappear. That is, for any \( i \in \{1, \ldots, L\} \), \( \Box(O_{i,1} \implies \Box(O_{i,1})) \) or equivalently,

\[
\Box(O_{i,1}) \implies \Box(O_{i,1}) \quad (16)
\]

We define \( \Box \varphi \) in (6) to be the conjunction of formula (13)–(16). (Note that for any LTL formulas \( \varphi \) and \( \psi \), \( \Box \varphi \land \Box \psi \) is equivalent to \( \Box(\varphi \land \psi) \)).

Next, we define the desired safety property, \( \Box \varphi_g \), as the conjunction of the following properties:

1) No collision. That is, for any \( i \in \{1, \ldots, L\} \) and \( j \in \{1, 2\} \),

\[
\Box(O_{i,j} \implies (x \in C_{x,i} \land y \in C_{y,j})) \quad (17)
\]
2) The vehicle stays in the right lane unless there is an obstacle blocking the lane. That is, for any \( i \in \{1, \ldots, L\} \),
\[
\bigcap \{O_{i,1} \land x \in C_{x,i} \} \implies (y \in C_{y,i})
\]
Finally, we define \( \varphi_y = (x \in C_{x,L}) \), i.e., we want to ensure that eventually the vehicle gets to the end of the road.

B. State Space Discretization

Since the dynamics and the constraints on the control efforts for the \( x \) and \( y \) components of the vehicle state are decoupled, we apply the discretization algorithm presented in Section IV for the \( x \) and \( y \) components separately for the sake of computational efficiency.\(^4\) Since the vehicle dynamics (8) are translationally invariant, we can use similar partition for all \( C_{z,i} \). The discretization algorithm with horizon length \( N = 10 \) and \( \text{Vol}_{\min} = 0.1 \) yields a partition with 11 cells \( \{C_{1,i}^{z}, C_{2,i}^{z}, \ldots, C_{11,i}^{z}\} \) for each \( C_{z,i} \) as shown in Fig. 3. For each \( i \in \{z_{\min} + 1, \ldots, z_{\max}\} \) and \( j \in \{1, \ldots, 11\} \), let \( C_{i,j}^{z} \) be the state label of cell \( C_{j,i}^{z} \) and let \( C_{z,i} = \{C_{1,i}^{z}, \ldots, C_{11,i}^{z}\} \). A discrete state is therefore a tuple \((\nu_{x}, \nu_{y}, O_{1,1}, \ldots, O_{L,2})\) where \((\nu_{x}, \nu_{y}) \in C_{x,i} \times C_{y,i}\) is the discrete controlled state. Using MPT [4], the reachability between discrete controlled states can be determined and a controller associated with each reachable pair of them can be generated such that the resulting continuous execution implements the discrete transition between them. The specification of the resulting finite transition system can then be derived as discussed in Section IV-C.

\[\text{Fig. 3. The partition of each cell } C_{z,i} \text{ in the original partition of the domain } C_{z}.\]

C. Receding Horizon Formulation

Based on the new partition of the vehicle state space, there are the total of \( 242 \times L \) discrete vehicle states and \( 2^{2xL} \) discrete environment states. Thus, in the worst case, the resulting automaton may have as many as \( 242 \times L \times 2^{2xL} \) nodes. To avoid the state explosion problem, we apply the receding horizon strategy on the variable \( x \). The partial order structure is defined as \( \Psi_{i} = \{(\nu_{x}, \nu_{y}, O_{1,1}, \ldots, O_{L,2}) \mid \nu_{x} \in C_{x,i}\} \). It can be easily shown that for any \( i < j \), \( W_{i} \prec_{\Psi} W_{j} \) since to avoid cell \( C_{x,L} \) starting from \( C_{x,i} \), the system needs to visit cell \( C_{x,j} \) for any \( j > i \).

Next, we need to define an invariant \( \Phi \) such that the specification (6) is realizable. Similar to \( z \), we let \( Z \) represent either \( X \) or \( Y \). We classify each component of the discrete controlled states into one of the following sets.

\[\text{a) A set } Z_{\text{notrans}} \text{ defined as: for any } \nu_{z} \in Z_{\text{notrans}}, i \in \{z_{\min} + 1, \ldots, z_{\max}\} \text{ and } j \in \{1, \ldots, 11\}, \nu_{z} \Rightarrow C_{z,i}^{j}.\]
\[\text{b) A set } Z_{\text{right}} \text{ defined as: for any } \nu_{z} \in Z_{\text{right}} \text{ such that } \nu_{z} \in C_{z,i}, \text{ there exists } k \in \{1, \ldots, 11\} \text{ such that } \nu_{z} \Rightarrow C_{z,i}^{k}.\]
\[\text{c) A set } Z_{\text{left}} \text{ defined as: for any } \nu_{z} \in Z_{\text{left}} \text{ such that } \nu_{z} \in C_{z,i}, \text{ there exists } k \in \{1, \ldots, 11\} \text{ such that } \nu_{z} \Rightarrow C_{z,i}^{k}.
\]
\[\text{d) A set } Z_{\text{same}} \text{ defined as: for any } \nu_{z} \in Z_{\text{same}} \text{ such that } \nu_{z} \in C_{z,i}, \text{ there exists } k \in \{1, \ldots, 11\} \text{ such that } \nu_{z} \Rightarrow C_{z,i}^{k}.
\]
\[\text{e) A set } Z_{both} \text{ defined as: for any } \nu_{z} \in Z_{both} \text{ such that } \nu_{z} \in C_{z,i}, \text{ there exists } k \in \{1, \ldots, 11\} \text{ such that } \nu_{z} \Rightarrow C_{z,i}^{k}.
\]
To restrict the initial states of the system assumed by \( A_{i} \), so that \( \Psi_{i} \) is realizable, we make the following observations.

1) To ensure the progress property \( \Diamond \varphi_{y} \), we need to assume that \( \nu_{z} \notin X_{\text{notrans}} \) and \( \nu_{y} \notin Y_{\text{notrans}} \).

2) To ensure no collision, the vehicle cannot collide with an obstacle at the initial state.

3) Suppose \( \nu_{x} \in C_{z,i} \). To ensure no collision, if \( \nu_{y} \) can only transition to \( \nu'_{x} \in C_{x,i} \), then either \( O_{i,1} \) is False or \( O_{i+1,1} \) is False. Similarly, if \( \nu_{y} \) can only transition to \( \nu'_{y} \in C_{y,2} \), then either \( O_{i,2} \) is False or \( O_{i+1,2} \) is False.

4) Suppose \( \nu_{x} \in C_{x,i} \) such that it can only transition to \( \nu'_{x} \in C_{x,i+1} \). To ensure no collision, if \( \nu_{y} \) can only transition to \( \nu'_{y} \in C_{y,2} \), then \( O_{i+1,1} \) is False. Similarly, if \( \nu_{y} \) can only transition to \( \nu'_{y} \in C_{y,2} \), then \( O_{i+1,2} \) is False. Similar observation can be derived for the case where \( \nu_{x} \in C_{x,i} \) such that it can only transition to \( \nu'_{y} \in C_{y,i} \).

5) To ensure the stay-in-lane property, the vehicle cannot be in the left lane unless there is an obstacle blocking the right lane at the initial state. In addition, the vehicle is never in the state \( (\nu_{x}, \nu_{y}) \in C_{x,i} \times C_{y,1} \) which can only transition to \( (\nu'_{x}, \nu'_{y}) \in C_{x,i} \times C_{y,2} \).

6) Suppose \( \nu_{x} \in C_{x,i} \) and \( O_{i+1,1} \) is False. To ensure that the vehicle does not go to the left lane when the right lane is not blocked, it is not the case that \( \nu_{y} \in C_{y,1} \) which can only transition to \( \nu'_{y} \in C_{y,2} \). In addition, it is not the case that \( \nu_{y} \) can only transition to \( \nu'_{x} \in C_{x,i+1} \) and \( \nu_{y} \in C_{y,2} \) which can only transition to \( \nu'_{y} \in C_{y,2} \).

We thus define \( \Phi \) to be the conjunction of all the above statements which can be expressed in LTL. For example, the corresponding LTL formula for observation 3 is a conjunction of the following formulas:

- \( (y \in Y_{\text{same}} \cap C_{y,i}) \lor (y \in Y_{\text{left}} \cap C_{y,2}) \land (x \in C_{x,i}) \implies (O_{i,1} \lor O_{i+1,1}) \)
- \( (y \in Y_{\text{right}} \cap C_{y,i}) \lor (y \in Y_{\text{same}} \cap C_{y,i}) \land (x \in C_{x,i}) \implies (O_{i,2} \lor O_{i+1,2}) \)

where \( Y_{\text{same}} = \bigcup_{i,j \in \mathbb{N}} C_{j,i}^{y_{\text{same}}} \), \( Y_{\text{left}} = \bigcup_{i,j \in \mathbb{N}} C_{j,i}^{y_{\text{left}}} \), and \( Y_{\text{right}} = \bigcup_{i,j \in \mathbb{N}} C_{j,i}^{y_{\text{right}}} \).

With \( d_{\text{pop}} = 1 \) and the horizon length \( 2 \) (i.e. \( g_{i} = i + 2 \)), the specification (7) is realizable. In addition, if we let \( d_{\text{obs}} \) be greater than 1 and restrict the initial state of the system such that \( \nu_{x} \notin X_{\text{notrans}} \) and \( \nu_{y} \notin Y_{\text{notrans}} \), we get that \( \varphi_{\text{init}} \equiv \Phi \) is a tautology.
D. Results

The synthesis was performed on a Pentium 4, 3.4 GHz computer with 4 Gb of memory. The computation time was 1230 seconds. The resulting automaton contains 2845 nodes. During the synthesis process, 96796 nodes were generated. Based on the authors experience, this particular computer crashes when approximately 97500 nodes are generated. Thus, this problem with horizon length 2 is as large as what the computer can handle. This means that without the receding horizon strategy, problems with the road of length greater than 3 cannot be solved.

A simulation result with the road length of 30 is shown in Fig. 4. The polygons drawn in red are the obstacles. Notice that when there is no obstacle blocking the lane, the vehicle tries to stay as close to the lane boundary \((y = 1)\) as possible. This is expected since to be able to avoid a pop up obstacle, due to the constraint on the admissible control inputs, the vehicle needs to stay close to the lane boundary to be able to change lane. To force the vehicle to stay close to the center of the lane, we need a finer partition of the road and extra LTL formula to ensure this property needs to be added to the system specification.

![Fig. 4. Simulation result. The polygons are the obstacles.](image)

VII. CONCLUSIONS AND FUTURE WORK

This paper illustrated how off-the-shelf tools from computer science and control can be integrated to allow automatic synthesis of complex dynamical systems which are guaranteed, by construction, to satisfy the desired properties expressed in temporal logic even in the presence of adversary. A receding horizon scheme was described that addresses the main limitation of the synthesis algorithm, the state explosion problem, and allows more complex problems to be solved, assuming that the system has a certain partial order structure. The example showed that without the receding horizon scheme, the synthesis problem can be extremely computationally challenging.

Although the adversarial nature of the environment has been incorporated in the synthesis, the effects of disturbances and modelling errors have not yet been studied. To increase the robustness of the system, we plan to impose more conditions on the multiparametric programming problem so that the continuous control law can be executed in a closed loop manner. In addition, the system specification needs to be modified to allow the possibility that the system may deviate from the plan due to disturbances and modelling errors.

Automatic or semi-automatic computation of an invariant \(\Phi\) in the receding horizon scheme based on the information provided by the synthesis tool is also of interest. This direction sounds promising since as described in the paper, \(\Phi\) can be constructed by iteratively adding, until the specification is realizable, a propositional formula which describes the initial state of the system for which there exists a set of moves of the environment such that the system cannot satisfy \(\Psi\).

VIII. ACKNOWLEDGMENTS

The authors gratefully acknowledge Hadas Kress-Gazit for the inspiring discussions on the automatic synthesis of finite state automata, for her suggestions and help with the synthesis tool, and for the code for generating an input to the synthesis tool and extracting its output; K. Mani Chandy and Knot Pipatsrisawat for the discussions and comments regarding the restriction of domain for the synthesis problem; and Vanessa Jönsson for thoughtful comments on the paper.

REFERENCES