Controller Synthesis for Linear System With Reach-Avoid Specifications

Chuchu Fan, Zengyi Qin, Umang Mathur, Qiang Ning, Sayan Mitra, and Mahesh Viswanathan

***Abstract*—We address the problem of synthesizing provably correct controllers for linear systems with reach-avoid specifications. Discrete abstraction-based controller synthesis techniques have been developed for linear and nonlinear systems with various types of specifications. However, these methods typically suffer from the state space explosion problem. Our solution decomposes the overall synthesis problem into two smaller, and more tractable problems: one synthesis problem for an open-loop controller, which can produce a reference trajectory, and a second for synthesizing a tracking controller, which can enforce the other trajectories to follow the reference trajectory. As a key building-block result, we show that, once a tracking controller is fixed, the reachable states from an initial neighborhood, subject to any disturbance, can be overapproximated by a sequence of ellipsoids, with shapes that are independent of the open-loop controller. Hence, the open-loop controller can be synthesized independently to meet the reach-avoid specification for an initial neighborhood. Moreover, we are able to reduce the problem of synthesizing open-loop controllers to satisfiability problems over quantifier-free linear real arithmetic. The number of linear constraints in the satisfiability problem is linear to the number of hyperplanes as the surfaces of the polytopic obstacles and goal sets. The overall synthesis algorithm, computes a tracking controller, and then iteratively covers the entire initial set to find open-loop controllers for initial neighborhoods. The algorithm is sound and, for a class of robust systems, is also complete. We implement this synthesis algorithm in a tool REALSYN VER 2.0 and use it on several benchmarks with up to 20 dimensions. Experiment results are very promising: REALSYN VER 2.0 can find controllers for most of the benchmarks in seconds.**

***Index Terms*—Controller synthesis, disturbance, linear system, reach-avoid specification.**

I. INTRODUCTION

THE controller synthesis question asks whether an input can be generated for a given system (or a plant) so that it achieves a given specification. Algorithms for answering this question hold the promise of automating controller design. They have the potential to yield high-assurance systems that are correct-by-construction, and even negative answers to the question can convey insights about unrealizability of specifications. This is neither a new nor a solved problem, but there has been resurgence of interest with availability of powerful tools like convex optimizations and satisfiability modulo theories (SMT) solvers, and compelling applications such as path planning [1], motion control [2], [3], and circuits design [4].

In this article, we study the control synthesis problem for linear, discrete-time, and time-varying plant models with bounded disturbance [5], [6]. We will consider *reach-avoid* specifications, which require that starting from any initial state Θ, the controller has to drive the system to a target set G, while avoiding certain unsafe states or obstacles $\mathbf{O}$. *Reach-avoid* specifications arise naturally in many domains such as autonomous and assisted driving, multirobot coordination, and spacecraft autonomy, and have been studied for linear, nonlinear, as well as stochastic models [7]–[12].

Textbook control design methods address specifications like stability, disturbance rejection, and asymptotic convergence, but they do not directly provide formal guarantees about reach-avoid specifications. Receding horizon control and model predictive control (MPC), have been broadly used on constrained control problems. Using MPC for reach-avoid specifications typically solves a sequence of mixed integer linear programming (MILP) [13], [14] or general nonlinear optimization problems [15], [16]. Another approach is based on *discrete abstractions*, where a discrete, finite-state, abstraction of the original control system is computed, and a discrete controller is synthesized by solving a two-player game on the abstracted game graph [17], [18]. Theoretically, these methods can be applied to systems with nonlinear dynamics and they can synthesize controllers for a general class of linear temporal logic (LTL) specifications. However, in practice, the discretization step leads

Manuscript received January 17, 2021; accepted March 4, 2021. Date of publication March 30, 2021; date of current version March 29, 2022. This work was supported in part by the DARPA Assured Autonomy under Grant FA8750-19-C-0089, in part by the National Science Foundation under Grant CCF 1901069, Grant CCF 2007428, and Grant FMITF 1918531, and in part by the Air Force Office of Scientific Research under Grant FA9550-17-1-0236. Recommended by Associate Editor L. Palopoli. *(Corresponding author: Chuchu Fan.)*

Chuchu Fan and Zengyi Qin are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: chuchu@mit.edu; qinzy@mit.edu).

Umang Mathur and Mahesh Viswanathan are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: umathur3@illinois.edu; vmahesh@illinois.edu).

Qiang Ning is with the Amazon, Inc., Boston, MA 02111 USA (e-mail: qiangning1990@gmail.com).

Sayan Mitra is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: mitras@illinois.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2021.3069723.

Digital Object Identifier 10.1109/TAC.2021.3069723

to state-space explosion for higher dimensional models. A detailed comparison between these methods and our proposed approach is provided in Section II.

In this article, the synthesis algorithm follows a natural paradigm for designing controllers. The approach is to separate the controller into two parts: an open-loop controller and a tracking controller, and synthesize them separately. An *open-loop* controller for a single initial state $x_0 \in \Theta$ to meet the reach-avoid specification. This is called the *reference trajectory*. For the remaining states in the initial set, a *tracking controller* is added, that drives these other trajectories towards the reference trajectory that starts from x_0. However, designing such a combined controller can be computationally expensive [19] because of the interdependency between the open-loop controller and the tracking controller (see Section IV-A). Our approach to making this construction feasible, is to demonstrate that the two controllers can be synthesized in a decoupled way as follows. We first design a tracking controller using a standard linear quadratic regulator (LQR) method [20]. The crucial result (see Lemma 1) that helps decouple the synthesis of the tracking and open-loop controller, is that for such a combined controller, once the tracking controller is fixed, the set of states reached from the initial set is contained within a sequence of ellipsoidal sets [21] centered around the reference trajectory. The shape and size of these ellipsoidal sets are solely dependent on the tracking controller and the disturbance, and are independent of the reference trajectory or the open-loop controller. In fact, this is a special case of constructing a Lyapunuv function for the error dynamics between the actual trajectory of the system and the reference trajectory [22]. Moreover, ellipsoids have been widely used in reachability computation to solve verification [23], [24] and synthesis [25] problems. In this article, we follow such controller design paradigm and enjoy the benefit of using ellipsoidal reachable sets: The open-loop controller and the resulting reference trajectory can be chosen independent of the fixed tracking controller.

Based on this, the problem of synthesizing the open-loop controller can be completely decoupled from synthesizing the tracking controller. Our open-loop controller is synthesized by encoding the problem as an SMT problem. The straightforward encoding of the synthesis problem is to find an open loop controller that can make sure all states in the reach set ellipsoids satisfy the reach-avoid specification. Such encoding results in a $\exists\forall$ formula in the theory of linear arithmetic. Unfortunately, solving large instances of such formulas using current SMT solvers is challenging. To overcome this, we exploit geometric properties of polytopes and ellipsoids, and reduce the original $\exists\forall$-formula into the quantifier-free fragment of linear arithmetic (`QF-LRA`). Moreover, assuming that the obstacles and goal set can be represented as polytopes, then the number of linear constraints in the `QF-LRA` formulas grows linearly with time and the number of hyperplanes as the surfaces in obstacles and the goal set (see Lemmas 2 and 3). In this way, the proposed approach for synthesizing the combined controller can scale to large dimensional systems.

Our overall algorithm (see Algorithm 1), after computing an initial tracking controller, iteratively synthesizes open-loop controllers by solving `QF-LRA` formulas for smaller subsets that cover the initial set. The algorithm will automatically identify the set of initial states for which the combined tracking+open-loop controller is guaranteed to work. Our algorithm is sound (see Theorem 1), and for a class of robust linear systems, it is also complete (see Theorem 2).

We have implemented the new synthesis algorithm in the tool REALSYN VER 2.0, which was developed with [26]. We compare the performance of the new algorithm proposed in this article with the previous algorithm as in [26], and a state-of-the-art synthesis tool SMC [27], [28], on 10 benchmark problems. Here, the obstacles are general polytopes instead of only axis-aligned hyper-rectangles. In REALSYN VER 2.0, any SMT solver can be plugged in for solving the synthesis problem. We report the results of using the `Yices` solver, as it outperformed other solvers in [26]. Results show that our new approach can achieve a 2 to 150 $\times$ speedup for most benchmark models comparing with the previous algorithm REALSYN VER 1.0 as in [26], and a 2–80 $\times$ speedup comparing with SMC. The proposed new algorithm also scales well for complex models—including a system with three vehicles (12-D) trying to reach a common goal while avoiding collision with the obstacles and each other, and another system with 10 vehicles (20-D) trying to maintain a platoon. For all the benchmark models, REALSYN VER 2.0 with the new algorithm finds a controller within 2 min using the `Yices` solver, and for most benchmarks it finds a controller within 10 s.

The major contributions of this article is to explore an assembly of several techniques from control, geometry, SAT solving to develop a fast and formally guaranteed algorithm for controller synthesis. To be more concrete, the following conditions hold.

1) We propose a synthesis algorithm to find correct-by-construction controllers for linear time-varying systems with respect to reach-avoid specifications. Our synthesis algorithm is sound, and is also complete for a class of robust linear systems.
2) Our proposed algorithm achieves scalability by reducing the synthesis problem to satisfiability over quantifier-free linear arithmetic and leveraging modern SMT solvers. We develop efficient encoding methods so that the number of constraints in the resulting SMT problem grows linearly with time and the complexity of the reach-avoid specification.
3) Our algorithm significantly improves the practical efficiency of control synthesis for large linear systems with disturbances. Empirical results show a significant improvement over state-of-the-art synthesis methods.

II. RELATED WORKS

Controller synthesis techniques have been the center of extensive investigation with numerous publications every year lately. Here, we briefly review related works based on different plant models, specifications, and several major approaches.

1) Models and Specifications for Synthesis: In increasing order of generality, the types of plant models that have been considered for controller synthesis are double-integrator models [2], linear dynamical models [13], [29]–[34], piecewise

affine models [8], [35], and nonlinear (possibly switched) models [7], [16], [17], [36], [37]. There is also a line of work on synthesis approaches for stochastic plants (see [38], and the references therein). For each of the classes, both continuous and discrete-time models have been addressed with possibly different approaches.

There are several classes of specifications typically used for synthesis as follows:

1) stabilization for system with special properties, including positive systems [29] and systems with quantized measurements [39], [40];
2) pure safety or invariance specifications [17], [41], [42];
3) reach-avoid [7]–[9], [17], [41];
4) general LTL, GR(1) [30], [43], [44] [31], [35], [45], metric temporal logic [46], and signal temporal logic [14], [47].

For each of these classes both bounded and unbounded-time variants have been considered.

In this article, we focus on linear, discrete-time, time-varying systems with reach-avoid specifications.

2) Model Predictive Control: MPC [48] utilizes an explicit plant model to predict the plant state and compute the control input to the plant based on this prediction. At each control interval, an MPC algorithm attempts to solve a constrained, discrete-time, optimal control problem in an online setting, with the objective of optimizing future plant behavior based on current state. Without loss of generality, assume the current state of the system is $\mathbf{x}[0]$, MPC solves a finite horizon (N steps) optimal control problem defined by

$$\begin{array}{ll} \min_{\mathbf{u}[0],\ldots,\mathbf{u}[N-1]} & V_f(\mathbf{x}[N]) + \sum_{i=0}^{N-1} \ell(\mathbf{x}[i], \mathbf{u}[i]) \\ \text{s.t.} & \bigwedge_{i=0}^{N} \mathbf{x}[i] \in X, \bigwedge_{j=0}^{N-1} \mathbf{u}[j] \in U \end{array} \tag{1}$$

where in the objective function V_f defines cost of the final state of the controlled system $\mathbf{x}[N]$, ℓ defines the cost of the rest of the states and control inputs, and the controlled system is required to satisfy the state and control constraints $\mathbf{x}[i] \in X, \mathbf{u}[i] \in U$, respectively. The *implicit* MPC law asks that at the state $\mathbf{x}[0]$, the first control $\mathbf{u}[0]$ of the computed optimal control sequence is applied, and the entire calculation is repeated at subsequent control intervals. When optimal control problems admit an explicit offline solution, online operations reduce to a simple function evaluation. Such *explicit* MPC has been exploited in many applications including motion planing [13], [33], [34]. The idea of explicit MPC is to solve the optimization problem (1) offline for all x within a given set, and to make the dependence of $u(t)$ on $x(t)$ explicit. The resulting MPC control law is a piecewise affine function of the state x defined over a polyhedral partition of the feasible set X_f. For systems with large state and input spaces, explicit MPC is not practical. Furthermore, it is hard to make explicit MPC handle cases where the system, cost function, or constraints are time-varying [49].

Using MPC for controller synthesis typically requires model reduction for casting the optimization problem (1) as a linear programming (LP) [33], quadratic programming (QP) [50], MILP [13], [14], [47], or general nonlinear optimization problems [15], [16].

In this article, the obstacles at each step are specified by a collection of polytopes. Therefore, the safe region X, as the complement of the obstacles, is usually nonconvex. To encode such avoidance condition $\mathbf{x}[i] \in X$ in the optimization problem (1), one has to introduce disjunctions to the constraints. Liu *et al.* [16] used Farkas' lemma to change the avoidance condition into its dual form that is compatible for MPC formulation. However, the extra variables introduced by Farkas' lemma will lead to nonlinear constraints. Vitus *et al.* [13] introduced extra Boolean variables to eliminate the disjunctions, and make the original optimization problem (1) an MILP. Both the works use implicit MPC law. The main drawback of implicit MPC is the need to solve a mathematical program online or within the sampling time to compute the control action. Therefore, it is hard to use on systems with large dimensionality [51] and when the sampling period is short. Explicit MPC can help relieve the heavy computation load, especially when the optimization problem is a LP or QP. However, in this case, the explicit solution for nonlinear optimization and MILP cannot be solved very efficiently in practice [51].

Compared with the above encoding for reach-avoid, our proposed method benefits from the fact that the tracking controller can fix the shapes and sizes of the reach set ellipsoids from an initial set. We further exploit special properties of the separation between ellipsoids and polytopes to make the constraints quantifier-free over linear real arithmetics, which can be efficiently solved using state-of-the-art SMT solvers or MILP solvers.

The major differences between our approach and the MPC-based approaches include the following.

1) Our approach does not require the help of a cost function. Instead, we only need a feasible solution of the satisfiability problem and sacrificed optimality.
2) MPC can be used in scenarios when obstacles are constructed dynamically when system evolves by solving the optimization problem (1) iteratively, while our proposed approach solves a one-shot SAT problem to find controllers that work for an initial set X_0 when obstacles are fixed. In Section IV-D, we discuss how to adjust our proposed encoding for the reach-avoid and input constraints to be used in MPC.

3) Control Lyapunov and Barrier Functions: The idea of control Lyapunov function (CLF) [52]–[54] is to associate a Lyapunov function $V(x)$ with its global minimum at the target state x^* to the nonlinear system that needs to be stabilized. At each time step, find a control input u to force $V(x)$ to decrease to guarantee that the target state x^* can be reached asymptotically.

Control barrier functions (CBF) [55] play a similar role to CLF in the study of liveness properties for nonlinear systems. CBF can ensure safety by enforcing invariance of a set. That is, CBF makes sure that that there exists a control input u such that the nonlinear system will not leave a safe set. In general, it is not easy to find a CLF for CBF for a given system.

4) Discrete Abstractions: Controller synthesis based on *discrete abstractions* have received considerable attention [17], [30], [30]–[32], [41], [43], [44]. These techniques involve constructing a finite partition of the continuous state space with respect to a set-valued map. Following those methods, it is possible to synthesize controllers for general nonlinear systems to enforce complex temporal logic specifications.

There is a growing set of controller synthesis tools and libraries based on the discrete abstraction approach. These include tools like CoSyMA [56], Pessoa [4], LTLMop [57], [58], Tulip [44], [59], and SCOTS [36]. Compared with these methods, our proposed solution takes a different route by "designing" the shape of reach sets first with the tracking controller, then "placing" the reach sets using the open loop controller. The entire process does not involve any partition of the state space, and therefore, avoids the potential problem of exponentially growing partitions for large dimensional systems. Our trial with a four-dimensional (4-D) example on Tulip [44], [59] did not finish the discretization step in one hour. Recent methods like feedback refinement [60] and multilayered abstraction [61] have been introduced to address the issue of exponentially growing partitions. However, such methods are yet to be available as synthesis tools. LTLMop [57], [58] handles GR(1) LTL specifications, which are more general than reach-avoid specifications considered in this article, but it is designed for 2-D robot models working in the Euclidean plane. It generates a hybrid controller as a combination of discrete controllers and continuous controllers to meet the high-level specification under certain assumptions on the environment.

5) Sampling-Based Path Planning: Sampling-based methods such as probabilistic road maps [62], rapidly-exploring random trees (RRT) [63], and fast marching tree [64] have offered the benefits of generating feasible trajectories through known or partially known environments. Compared with the deterministic guarantees provided by synthesis methods discussed previously, including ours, the sampling based methods come with stochastic guarantees. Also, they are not designed to be robust to model uncertainty or disturbances.

6) Satisfiability Modulo Convex Optimization: SMC [27], [28] solves satisfiability problems, which are represented as Boolean combinations of convex constraints over the real numbers. Unlike our approach that reduces the reach-avoid problem to a pure SAT problem, SMC uses a combination of SAT solving and convex programming to provide a satisfying assignment or determine that the formula is unsatisfiable. Therefore, SMC enjoys both the efficiency of convex optimizations and the formal guarantees of SAT solving, while our approach depends more on the efficiency of SMT solvers over quantifier-free linear real arithmetic. SMC can be used to solve robotic motion planning problems and has been shown to be much more effective than sampling-based methods like RRT. In Section V, we compare our proposed algorithm with SMC by adapting the original implementation of SMC to handle our examples.

In addition to the abovementioned approaches, an alternative synthesis technique generates mode switching sequences for switched system models [65]–[69] to meet the specifications. This line of work focuses on a finite input space, instead of the infinite input space we are considering in this articles.

Abate *et al.* [42] used a controller template similar to the one considered in this article for invariant specifications. A counter-example guided inductive synthesis approach is used to first find a feedback controller for stabilizing the system. Since this feedback controller may not be safe for all initial states of the system, a separate verification step is employed to verify safety, or alternatively to find a counter example. In the latter case, the process is repeated until a valid controller is found. This is different from our approach, where any controller found needs no further verification.

III. Preliminaries and Problem Statement

A. Notations

For a set S and a finite or infinite sequence σ of elements from S, we denote the tth element of σ by $\sigma[t]$. In the rest of the article, we will use boldfaced letters (for example, $\mathbf{A}, \mathbf{B}, \mathbf{x}, \mathbf{d}, \mathbf{u}$, etc.,) to denote a sequence of matrices or vectors. Given a vector $x \in \mathbb{R}^n$, $x(i)$ is the ith component of x. Given a matrix $A \in \mathbb{R}^{n\times m}$, $A^{(i)}$ is the ith row of A. Given an invertible matrix $M \in \mathbb{R}^{n\times n}$ and a vector $x \in \mathbb{R}^n$, $\|x\|_M \triangleq \sqrt{x^\top M^\top M x}$ is called the M-*norm* of x.

Given a vector $c \in \mathbb{R}^n$, an invertible matrix M, and a scalar value $r \geq 0$, we define $E_r(c, M) \triangleq \{x \mid \|x - c\|_M \leq r\}$ to be the ellipsoid centered at c with radius r and shape M. $B_r(c) \triangleq E_r(c, I)$ is the ball of radius r centered at c. For two sets $R, S \subseteq \mathbb{R}^n$, we define $R \oplus S \triangleq \{x + y \mid x \in R, y \in S\}$; for a singleton set, we abuse notation and use $v \oplus S$ to denote $\{v\} \oplus S$. For set $S \subseteq \mathbb{R}^n$ and matrix $M \in \mathbb{R}^{n\times n}$, we define $M \otimes S \triangleq \{Mx \mid x \in S\}$. We say a set $S \subseteq \mathbb{R}^n$ is a polytope if there is a matrix $A^{k\times n}$ and a vector $b \in \mathbb{R}^k$ such that $S = \{x \mid Ax \leq b\}$.

B. Discrete Time Linear Control Systems

An (n, m)-*dimensional time-varying discrete-time linear system*$\mathcal{A}$ is a five-tuple $\langle \mathbf{A}, \mathbf{B}, \Theta, U, D\rangle$, where the following conditions hold:

1) $\mathbf{A}$ is an infinite sequence of $\mathbb{R}^{n\times n}$ matrices, called *dynamic matrices*;
2) $\mathbf{B}$ is an infinite sequence of $\mathbb{R}^{n\times m}$ matrices, called *input matrices*, and at each time step t, the pair $(\mathbf{A}[t], \mathbf{B}[t])$ is controllable [6];
3) $\Theta \subseteq \mathbb{R}^n$ is a *set of initial states*;
4) $U \subseteq \mathbb{R}^m$ is the *space of inputs*;
5) $D \subseteq \mathbb{R}^n$ is the *space of disturbances*.

A *control sequence* for an (n, m)-dimensional system $\mathcal{A}$ is a (possibly infinite) sequence $\mathbf{u} = \mathbf{u}[0], \mathbf{u}[1], \ldots$, where each $\mathbf{u}[t] \in U$. Similarly, a *disturbance sequence* for $\mathcal{A}$ is a (possibly infinite) sequence $\mathbf{d} = \mathbf{d}[0], \mathbf{d}[1], \ldots$, where each $\mathbf{d}[t] \in D$. Given control $\mathbf{u}$ and disturbance $\mathbf{d}$, and an initial state $\mathbf{x}[0] \in \Theta$, the *execution of* $\mathcal{A}$ is uniquely defined as the (possibly infinite) sequence of states $\mathbf{x} = \mathbf{x}[0], \mathbf{x}[1], \ldots$, where for each $t > 0$

$$\mathbf{x}[t+1] = \mathbf{A}[t]\mathbf{x}[t] + \mathbf{B}[t]\mathbf{u}[t] + \mathbf{d}[t]. \tag{2}$$

An *open-loop control sequence* (also called an *open-loop controller*) for a given single initial state $x_0 \in \Theta$ is a control sequence $\mathbf{u}$ such that the corresponding execution $\mathbf{x}$ with $\mathbf{x}[0] = x_0$ and 0 disturbance (i.e., $\forall t \geq 0, \mathbf{d}[t] = 0$) satisfies the reach-avoid constraints..

A *(state feedback) controller* for $\mathcal{A}$ is a function $g : \Theta \times \mathbb{R}^n \to \mathbb{R}^m$, that maps an initial state and a (current) state to an input. That is, given an initial state $x_0 \in \Theta$ and state $x \in \mathbb{R}^n$

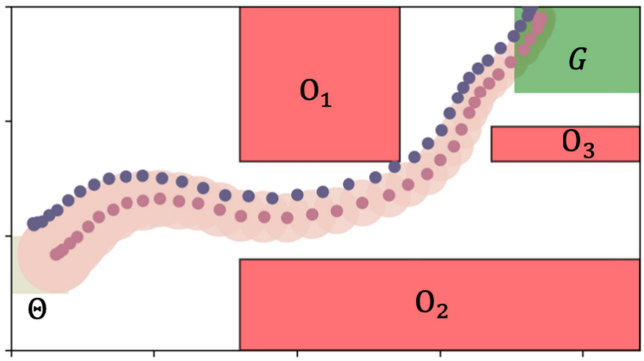

Fig. 1. Settings for controller synthesis of a mobile robot with reach-avoid specification.

at time t, the control input to the plant at time t is

$$\mathbf{u}[t] = g(x_0, x). \tag{3}$$

This controller is allowed to use the memory of some initial state x_0 (not necessarily the current execution's initial state) for deciding the current state-dependent feedback. Thus, given an initial state $\mathbf{x}[0]$, a disturbance $\mathbf{d}$, and a state feedback controller g, Equations (2) and (3) define a unique execution $\mathbf{x}$ of $\mathcal{A}$. A state x is *reachable at the* tth*-step* if there exists an execution $\mathbf{x}$ of $\mathcal{A}$ such that $\mathbf{x}[t] = x$. The set of all reachable states from some set $S \subseteq \Theta$ in exactly T steps using the controller g is denoted by $\mathsf{Reach}_{\mathcal{A},g}(S, T)$. When $\mathcal{A}$ and g are clear from the context, we simply write $\mathsf{Reach}(S, T)$.

C. Bounded Controller Synthesis Problem

Given an (n, m)*-dimensional time-varying discrete-time linear system* $\mathcal{A}$, a sequence $\mathbf{O}$ of *obstacles* or unsafe sets (with $\mathbf{O}[t] \subseteq \mathbb{R}^n$, for each t), a *goal* $G \subseteq \mathbb{R}^n$, and a time bound T, the *bounded time controller synthesis problem* is to find, a state feedback controller g such that for every initial state $\theta \in \Theta$ and any disturbance sequence $\mathbf{d} \in D^T$ of length T, the unique execution $\mathbf{x}$ of $\mathcal{A}$ with g, starting from $\mathbf{x}[0] = \theta$, satisfies the following:

1) for all $t \leq T$, $\mathbf{u}[t] \in U$;
2) for all $t \leq T$, $\mathbf{x}[t] \notin \mathbf{O}[t]$;
3) $\mathbf{x}[T] \in G$.

For the rest of the article, we will assume that each of the sets in $\{\mathbf{O}[t]\}_{t\in\mathbb{N}}$, G and U are closed polytopes.

The controller synthesis problem requires one to find a state feedback controller that ensures that the execution starting from any initial state in Θ will meet the reach-avoid specification. Since the set of initial states Θ will typically be an infinite set, this requires the synthesized feedback controller g to have a finite representation. An "enumerative" representation, where a (separate) *open-loop control sequence* is constructed for each initial state, is not feasible. We, therefore, need a useful template that will serve as the representation for the feedback controller.

Example 1: Consider a mobile robot that needs to reach the green area of an apartment starting from the entrance area, while avoiding the red areas (see Fig. 1). The robot's dynamics are described by a linear model (for example the navigation model from [70]). The obstacle sequence $\mathbf{O}$ (red rectangles and outside of the figure region) here is static, that is, $\mathbf{O}[t] = \mathbf{O}[0]$ for all $t \geq 0$. Both Θ (light green) and G (dark green) are rectangles (which are also polytopes). Although these sets are depicted in 2-D, the dynamics of the robot may involve a higher dimensional state space.

In this example, there is no disturbance, but a similar problem can be formulated for a drone flying outdoors, in which case, the disturbance input could model the effect of wind. Time-varying obstacle sets are useful for modeling safety requirements of multirobot systems.

Suppose robot is asked to reach the target set in 40 steps. The dotted curves are two executions from Θ and the pink ellipsoids show the projection of the reachset on the robot's position with a synthesized controller.

IV. Synthesis Algorithm

A. Algorithm Overview

A natural controller design paradigm is to first find a *reference execution* $\mathbf{x}_{\mathsf{ref}}$, which uses an *open-loop controller*, then add a *tracking controller*, which tries to force other executions $\mathbf{x}$ starting from different initial states $\mathbf{x}[0]$ to get close to $\mathbf{x}_{\mathsf{ref}}$ by minimizing the distance between $\mathbf{x}_{\mathsf{ref}}$ and $\mathbf{x}$. This form of controller combining open-loop control with tracking control is also proposed in [19] for reach-avoid specifications. For the discrete-time linear control system defined as (2), the combined controller is formally defined as follows.

Definition 1: Given a discrete-time linear system as (2), the combined controller g is a tuple $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ such that the control input $\mathbf{u}[t]$ to the system is

$$\mathbf{u}[t] = \mathbf{u}_{\mathsf{ref}}[t] + \mathbf{K}[t](\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t]), \text{with} \tag{4}$$

$$\mathbf{x}_{\mathsf{ref}}[t+1] = \mathbf{A}[t]\mathbf{x}_{\mathsf{ref}}[t] + \mathbf{B}[t]\mathbf{u}_{\mathsf{ref}}[t] \tag{5}$$

where

1) $\mathbf{u}_{\mathsf{ref}}$ is called the open-loop control sequence, which determines the value of the reference execution $\mathbf{x}_{\mathsf{ref}}[t]$ at each time step $t \in \mathbb{N}$ once $\mathbf{x}_{\mathsf{ref}}[0]$ is fixed,
2) $\mathbf{K}$ is called the tracking controller, which is a sequence of matrices that determine the additive component of the input based on the difference between the current state and the reference execution.

Given the combined feedback controller g as the tuple $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$, we could rewrite the linear system in (4) as an augmented system

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{x}_{\mathsf{ref}} \end{bmatrix}[t+1] = \begin{bmatrix} \mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t] & -\mathbf{B}[t]\mathbf{K}[t] \\ 0 & \mathbf{A}[t] \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_{\mathsf{ref}} \end{bmatrix}[t] + \begin{bmatrix} \mathbf{B}[t] & 0 \\ 0 & \mathbf{B}[t] \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\mathsf{ref}} \\ \mathbf{u}_{\mathsf{ref}} \end{bmatrix}[t] + \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}[t].$$

Observe that the above-mentioned augmented system has the form

$$\hat{\mathbf{x}}[t+1] = \hat{\mathbf{A}}[t]\hat{\mathbf{x}}[t] + \hat{\mathbf{B}}[t]\hat{\mathbf{u}}[t] + \hat{\mathbf{d}}[t]$$

and its closed-form solution is given by

$$\hat{\mathbf{x}}[t] = \left(\prod_{i=0}^{t-1} \hat{\mathbf{A}}[i]\right) \hat{\mathbf{x}}[0] + \sum_{i=0}^{t-1} \left(\prod_{j=i+1}^{t-1} \hat{\mathbf{A}}[j]\right) (\hat{\mathbf{B}}[i]\hat{\mathbf{u}}[i] + \hat{\mathbf{d}}[i]). \tag{6}$$

To synthesize a controller g of this form, therefore, requires finding $\mathbf{K}$, $\mathbf{x}_{\text{ref}}[0]$, $\mathbf{u}_{\text{ref}}$ such that the closed-form solution meets the reach-avoid specification. This is indeed the approach followed in [19], albeit in the continuous time setting. Observe that in the closed-form solution, $\hat{\mathbf{A}}[t]$, $\hat{\mathbf{u}}$, and $\hat{\mathbf{x}}[0]$ all depend on parameters that we need to synthesize. Therefore, solving such constraints involves polynomials whose degrees grow with the time bound. This is very expensive, and unlikely to scale to large dimensions and time bounds.

In this article, to achieve scalability, we take a slightly different approach than the one where $\mathbf{K}$, $\mathbf{x}_{\text{ref}}[0]$, and $\mathbf{u}_{\text{ref}}$ are simultaneously synthesized. We first synthesize a tracking controller $\mathbf{K}$, *independent* of $\mathbf{x}_{\text{ref}}[0]$ and $\mathbf{u}_{\text{ref}}$, using the standard LQR method. Once $\mathbf{K}$ is synthesized, we show that, no matter what $\mathbf{x}_{\text{ref}}[0]$ and $\mathbf{u}_{\text{ref}}$ are, the state of the system at time t starting from x_0 is guaranteed to be contained within an ellipsoid centered at $\mathbf{x}_{\text{ref}}[t]$ with shape and radius that depend only on $\mathbf{K}$, the initial distance between x_0 and $\mathbf{x}_{\text{ref}}[0]$, time t, and disturbance set D. Moreover, this radius is only a *linear* function of the initial distance (see Lemma 1). Thus, if we can synthesize an open-loop controller $\mathbf{u}_{\text{ref}}$ starting from some state $\mathbf{x}_{\text{ref}}[0]$, such that ellipsoids centered around $\mathbf{x}_{\text{ref}}$ satisfy the reach-avoid specification, we can conclude that the combined controller will work correctly for all initial states in some ball around the initial state $\mathbf{x}_{\text{ref}}[0]$. The radius of the ball around $\mathbf{x}_{\text{ref}}[0]$ for which the controller is guaranteed to work will depend on the radii of the ellipsoids around $\mathbf{x}_{\text{ref}}$ that satisfy the reach-avoid specification. This decoupled approach to synthesis is the first key idea in our algorithm.

Synthesizing the tracking controller $\mathbf{K}$ still leaves open the problem of synthesizing an open-loop controller for an initial state $\mathbf{x}_{\text{ref}}[0]$. A straightforward encoding of the problem could be to find an open-loop controller that works for all initial states in some ball around $\mathbf{x}_{\text{ref}}[0]$. That is, finding a satisfying solution for the formula $\exists \mathbf{u}_{\text{ref}}, \exists r$, such that $\forall \mathbf{x}[0] \in B_r(\mathbf{x}_{\text{ref}}[0]), \bigwedge_{t=0}^{T} \mathbf{x}[t] \notin \mathbf{O}[t] \wedge \mathbf{x}[T] \in G$. This results in a $\exists\forall$-formula in the theory of real arithmetic. Unfortunately, solving such formulas does not scale to large dimensional systems using current SMT solvers [71]. The next key idea in our algorithm is to simplify these constraints and make the formula quantifier free. We reduce the problem of deciding whether an ellipsoid (the set of reachable states) is separated from (or contained in) a polytope (the obstacles or the goal) to measuring the distances of the center of the ellipsoid to surfaces of the polytopes in a linearly transformed coordinate. In this way, we are able to reduce the original $\exists\forall$-formula into the `QF-LRA` [72], [73] (see Section IV-D).

Putting it all together, the overall algorithm (see Algorithm 1) works as follows. After computing an initial tracking controller $\mathbf{K}$, it synthesizes open-loop controllers for different initial states by solving `QF-LRA` formulas. After each open-loop controller is synthesized, the algorithm identifies the set of initial states for which the combined tracking+open-loop controller is guaranteed to work, and removes this set from Θ. In each new iteration, it picks a new initial state not covered by previous combined controllers, and the process terminates when all of Θ is covered. Our algorithm is sound (see Theorem 1)—whenever a controller is synthesized, it meets the specifications. Furthermore, for robust systems (defined later in this article), our algorithm is guaranteed to terminate when the system has a combined controller for all initial states (see Theorem 2).

B. Synthesizing the Tracking Controller $\mathbf{K}$

Given any open-loop controller $\mathbf{u}_{\text{ref}}$ and the corresponding reference execution $\mathbf{x}_{\text{ref}}$, by replacing in (2), the controller of (4), we get

$$\begin{aligned}\mathbf{x}[t+1] = {} & (\mathbf{A}[t] + \mathbf{B}[t]K[t])\,\mathbf{x}[t] - \mathbf{B}[t]\mathbf{K}[t]\mathbf{x}_{\text{ref}}[t] \\ & + \mathbf{B}[t]\mathbf{u}_{\text{ref}}[t] + \mathbf{d}[t].\end{aligned} \tag{7}$$

Subtracting $\mathbf{x}_{\text{ref}}[t+1]$ from both sides, we have that for any execution $\mathbf{x}$ starting from the initial states $\mathbf{x}[0]$ and with disturbance $\mathbf{d}$, the distance between $\mathbf{x}$ and $\mathbf{x}_{\text{ref}}$ changes with time as

$$\begin{aligned}&\mathbf{x}[t+1] - \mathbf{x}_{\text{ref}}[t+1] \\ &\quad = (\mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t])\,(\mathbf{x}[t] - \mathbf{x}_{\text{ref}}[t]) + \mathbf{d}[t].\end{aligned} \tag{8}$$

With $\mathbf{A}_c[t] \triangleq \mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t]$, $\mathbf{y}[t] \triangleq \mathbf{x}[t] - \mathbf{x}_{\text{ref}}[t]$, (8) becomes

$$\mathbf{y}[t+1] = \mathbf{A}_c[t]\mathbf{y}[t] + \mathbf{d}[t].$$

We want $\mathbf{x}[t]$ to be as close to $\mathbf{x}_{\text{ref}}[t]$ as possible, which means $\mathbf{K}[t]$ should be designed to make $|\mathbf{y}[t]|$ converge. Equivalently, $\mathbf{K}[t]$ should be designed as a linear feedback controller such that the system $\mathbf{y}[t+1] = \mathbf{A}_c[t]\mathbf{y}[t]$ is stable. Such a matrix $\mathbf{K}[t]$ can be computed using several methods. In this article, we compute $\mathbf{K}[t]$ as finding a linear state feedback controller by solving the LQR problem [20], stated as follows.

Definition 2 (LQR): For a time-varying linear system $\mathcal{A}$ as defined in Section III-B with 0 disturbance and a time bound T, the LQR problem is the optimal control problem of finding open loop control $\mathbf{u}[0], \ldots, \mathbf{u}[T-1]$, such that the following objective function is minimized:

$$\begin{aligned}J(\mathbf{x}[0], \mathbf{u}, T) \triangleq {} & \mathbf{x}[T]^\top \mathbf{Q}[T]\mathbf{x}[T] \\ & + \sum_{t=0}^{T-1} (\mathbf{x}[t]^\top \mathbf{Q}[t]\mathbf{x}[t] + \mathbf{u}[t]^\top \mathbf{R}[t]\mathbf{u}[t])\end{aligned}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are sequences of symmetric positive definite matrices.

The optimal control for LQR is given by $\forall t = 0, \ldots, T-1$, $\mathbf{u}[t] = \mathbf{K}[t]\mathbf{x}[t]$ where

$$\mathbf{K}[t] \triangleq -\left(\mathbf{B}[t]^\top \mathbf{P}[t+1]\mathbf{B}[t] + \mathbf{R}[t]\right)^{-1} \mathbf{B}[t]^\top \mathbf{P}[t+1]\mathbf{A}[t] \tag{9}$$

and $\mathbf{P}[t]$ is computed by solving the discrete time Riccati difference equation

$$\mathbf{P}[t] = \mathbf{A}[t]^\top \mathbf{P}[t+1]\mathbf{A}[t] + \mathbf{Q}[t] - \mathbf{A}[t]^\top \mathbf{P}[t+1]\mathbf{B}[t] \\ (\mathbf{B}[t]^\top \mathbf{P}[t+1]\mathbf{B}[t] + \mathbf{R}[t])^{-1}\mathbf{B}[t]^\top \mathbf{P}[t+1]\mathbf{A}[t]$$

with boundary condition $\mathbf{P}[T] = \mathbf{Q}[T]$ [74]. The matrices $\mathbf{K}$ in (9) can be used as a tracking controller as in Definition 1.

When $T \to \infty$ and $\forall t \geq 0, \mathbf{A}[t] = A, \mathbf{B}[t] = B, \mathbf{Q}[t] = Q$, and $\mathbf{R}[t] = R$ are all constant matrices, and $\mathbf{K}[t]$ computed using (9) will also become a constant matrix K. Furthermore, if the pair (A, B) is controllable (or stabilizable), the closed-loop system $\mathbf{x}[t+1] = (A + BK)\mathbf{x}[t]$ is stable. That is, the eigenvalues of $A_c = A + BK$ with K given by (9) have magnitudes less than 1. Therefore, when $T \to \infty$, the tracking controller K computed using LQR can guarantee that the any execution $\mathbf{x}$ will converge to $\mathbf{x}_{\mathsf{ref}}$ asymptotically when there is no disturbance.

For most of the experiments presented in Section V, we fix each $\mathbf{Q}[t]$ and $\mathbf{R}[t]$ to be identity matrices. Roughly, for a given $\mathbf{R}$, scaling up $\mathbf{Q}$ results in a $\mathbf{K}$ that makes an execution $\mathbf{x}$ converge faster to the reference execution $\mathbf{x}_{\mathsf{ref}}$ but will also result in larger values of $\mathbf{u}$. In this article, the detailed tradeoffs involved in the choices of $\mathbf{Q}[t]$ and $\mathbf{R}[t]$ will not be pursued further.

With the synthesized $\mathbf{K}$, we are able to compute the set of reachable states for $\mathcal{A}$ with an arbitrary reference trajectory $\mathbf{x}_{\mathsf{ref}}$, as shown in the following section.

C. Reachset Overapproximation With Tracking Controller

In this section, we assume that the tracking controller, which is a sequence of matrices $\mathbf{K}$, computed as in Section IV-B, will make $\mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t]$ invertible for any time t. We do not need $\mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t]$ to be stable for the analysis of the rest of the article. However, later on we will see that if $\mathbf{K}$ can make the other trajectories $\mathbf{x}$ converge to $\mathbf{x}_{\mathsf{ref}}$, the set of reachable states will also converge to its center $\mathbf{x}_{\mathsf{ref}}$, which is desirable for the overall synthesis algorithm.

Once we fix $\mathbf{K}$, we show that the reachable states of the system $\mathcal{A}$ with an open-loop controller $\mathbf{u}_{\mathsf{ref}}$ (to be computed in Section IV-D) can be overapproximated using a sequence of ellipsoids centered at the corresponding $\mathbf{x}_{\mathsf{ref}}$ with shapes and radii depending on $\mathbf{A}, \mathbf{B}, \mathbf{K}$, the initial set, and the disturbances (see Lemma 1). Moreover, for systems with 0 disturbances (i.e., $D = \{0\}$), Corollary 1 shows that the set of reachable states can be computed precisely (i.e., there is no overapproximation error).

Lemma 1: Consider a linear system $\mathcal{A} = \langle \mathbf{A}, \mathbf{B}, \Theta, U, D \rangle$ with a controller defined as in (4). Fix the following:

1) a tracking controller $\mathbf{K}$ such that $\mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t]$ is invertible for each time t;
2) an open-loop controller $\mathbf{u}_{\mathsf{ref}}$ with the corresponding reference execution $\mathbf{x}_{\mathsf{ref}}$;
3) an ellipsoidal initial set $S = E_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0], \mathbf{M}[0]) \subseteq \Theta$, where $\mathbf{r}[0]$ and $\mathbf{M}[0]$ are the radius and shape of the ellipsoid, respectively. Then,

$$\mathsf{Reach}(S, t) \subseteq E_{\mathbf{r}[t]}(\mathbf{x}_{\mathsf{ref}}[t], \mathbf{M}[t]), \forall\, t \leq T, \text{ where}$$

$$\mathbf{M}[t] = \mathbf{M}[0]\left(\prod_{i=0}^{t-1}(\mathbf{A}[i] + \mathbf{B}[i]\mathbf{K}[i])^{-1}\right)$$

$$\mathbf{r}[t] = \mathbf{r}[0] + \sum_{i=0}^{t-1}\delta[i], \tag{10}$$

$\delta[i]$ is chosen such that $\forall i \geq 0, E_{\delta[i]}(0, \mathbf{M}[i+1]) \supseteq D$.

Proof: We prove this lemma by induction on t.

Base case: When $t = 0$, from the condition (3) of the Lemma we know that $\mathsf{Reach}(S, 0) = S = E_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0], \mathbf{M}[0])$.

Induction step: Assume that at time step t, we have $\mathsf{Reach}(S, t) \subseteq E_{\mathbf{r}[t]}(\mathbf{x}_{\mathsf{ref}}[t], \mathbf{M}[t])$.

Let $\mathbf{A}_c[t] = \mathbf{A}[t] + \mathbf{B}[t]\mathbf{K}[t]$. At time step $t+1$, from (8), we have that

$$\mathbf{x}[t+1] = \mathbf{x}_{\mathsf{ref}}[t+1] + \mathbf{A}_c[t](\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t]) + \mathbf{d}[t].$$

$\forall \mathbf{x}[t] \in \mathsf{Reach}(S, t)$, we have $\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t] \in E_{\mathbf{r}[t]}(0, \mathbf{M}[t])$. Moreover, since $\mathbf{d}[t] \in D$, we have that

$$\mathbf{x}[t+1] \in \mathbf{x}_{\mathsf{ref}}[t+1] \oplus \mathbf{A}_c[t]E_{\mathbf{r}[t]}(0, \mathbf{M}[t]) \oplus D. \tag{11}$$

Recall that $\oplus$ is the addition of all elements of sets, and $\mathbf{A}_c[t]E_{\mathbf{r}[t]}(0, \mathbf{M}[t])$ means multiplying each vector in $E_{\mathbf{r}[t]}(0, \mathbf{M}[t])$ with $\mathbf{A}_c[t]$.

The right-hand side of (11) can be computed as follows.

1) The second item $\mathbf{A}_c[t]E_{\mathbf{r}[t]}(0, \mathbf{M}[t])$, which contains all possible values of $\mathbf{A}_c[t](\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t])$, can be computed as

$$\mathbf{A}_c[t]E_{\mathbf{r}[t]}(0, \mathbf{M}[t]) = \{\mathbf{A}_c[t]x \mid \|x\|_{\mathbf{M}[t]} \leq \mathbf{r}[t]\} \\ = \{\mathbf{A}_c[t]x \mid \|\mathbf{M}[t]x\|_2 \leq \mathbf{r}[t]\}.$$

Letting $y = \mathbf{A}_c[t]x$, then, we have

$$\mathbf{A}_c[t]E_{\mathbf{r}[t]}(0, \mathbf{M}[t]) = \{y \mid \|\mathbf{M}[t]\mathbf{A}_c^{-1}[t]y\|_2 \leq \mathbf{r}[t]\} \\ = \{y \mid \|y\|_{\mathbf{M}[t]\mathbf{A}_c[t]^{-1}} \leq \mathbf{r}[t]\} = E_{\mathbf{r}[t]}(0, \mathbf{M}[t+1]).$$

2) Then, since $D \subseteq E_{\delta[t]}(0, \mathbf{M}[t+1])$, which means $\forall d \in D$, $\|d\|_{\mathbf{M}[t+1]} \leq \delta[t]$. Therefore, we have

$$E_{\mathbf{r}[t]}(0, \mathbf{M}[t+1]) \oplus D \\ = \{x + d \mid \|x\|_{\mathbf{M}[t+1]} \leq \mathbf{r}[t], \|d\|_{\mathbf{M}[t+1]} \leq \delta[t]\}.$$

Using triangular inequality of the $M[t+1]$ norm, we have

$$E_{\mathbf{r}[t]}(0, \mathbf{M}[t+1]) \oplus D \subseteq \{y \mid \|y\|_{\mathbf{M}[t+1]} \leq \mathbf{r}[t] + \delta[t]\} \\ = E_{\mathbf{r}[t+1]}(0, \mathbf{M}[t+1]).$$

3) Finally, it is easy to see that

$$\mathbf{x}_{\mathsf{ref}}[t+1] \oplus E_{\mathbf{r}[t+1]}(0, \mathbf{M}[t+1]) \\ = E_{\mathbf{r}[t+1]}(\mathbf{x}_{\mathsf{ref}}[t+1], \mathbf{M}[t+1]).$$

Therefore, we have

$$\mathsf{Reach}(S, t+1) \subseteq E_{\mathbf{r}[t+1]}(\mathbf{x}_{\mathsf{ref}}[t+1], \mathbf{M}[t+1]).$$

■

In the above-mentioned proof, the only overapproximation happened in Step 2, as we overapproximate the disturbance D using an ellipsoid with shape $\mathbf{M}[t+1]$. This is because we want to keep reach sets represented as ellipsoids all the time. If there is no disturbance, i.e., $D = \{0\}$, we do not need to conduct Step 2, and Lemma 1 can give us exact reach sets.

Corollary 1: Consider a linear system $\mathcal{A} = \langle \mathbf{A}, \mathbf{B}, \Theta, U, D = \{0\} \rangle$ with a controller defined as in Equation (4). Fix the following:

1) a tracking controller $\mathbf{K}$;
2) an open-loop controller $\mathbf{u}_{\text{ref}}$ with the corresponding reference execution $\mathbf{x}_{\text{ref}}$;
3) an ellipsoidal initial set $S = E_{\mathbf{r}[0]}(\mathbf{x}_{\text{ref}}[0], \mathbf{M}[0]) \subseteq \Theta$, where $\mathbf{r}[0]$ and $\mathbf{M}[0]$ are the radius and shape of the ellipsoid respectively. Then,

$$\mathsf{Reach}(S, t) = E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t]), \forall\, t \leq T \tag{12}$$

where $\mathbf{M}[t] = \mathbf{M}[0](\prod_{i=0}^{t-1}(\mathbf{A}[i] + \mathbf{B}[i]\mathbf{K}[i])^{-1})$.

In Lemma 1, $\mathbf{r}[0]$ and $\mathbf{M}[0]$ can be chosen arbitrarily as long as the corresponding ellipsoid $E_{\mathbf{r}[0]}(\mathbf{x}_{\text{ref}}[0], \mathbf{M}[0])$ contains (or is equal to) the initial set S. It follows that given any sequence of $\mathbf{u}_{\text{ref}}$ as the open-loop controller, which leads to a corresponding reference trajectory $\mathbf{x}_{\text{ref}}$, the reachable states from S $\mathsf{Reach}(S, t)$ can be overapproximated by an ellipsoid centered at $\mathbf{x}_{\text{ref}}[t+1]$ with shape $\mathbf{M}[t] = \mathbf{M}[0](\prod_{i=0}^{t-1}(\mathbf{A}[i] + \mathbf{B}[i]\mathbf{K}[i])^{-1})$ and radius $\mathbf{r}[0]$ (when there is no disturbance) or $\mathbf{r}[0]$ plus an additive term $\sum_{i=0}^{t-1}\delta[i]$, which accounts for bounded disturbance. Note that the shapes and radii of the ellipsoids are all independent of the open-loop controller $\mathbf{u}_{\text{ref}}$ and the reference trajectory $\mathbf{x}_{\text{ref}}$. This is the key step to decouple the synthesis of the tracking controller $\mathbf{K}$ and rest of the parameters in the feedback controller ($\mathbf{u}_{\text{ref}}$, $\mathbf{x}_{\text{ref}}[0]$). In the following section, we discuss a novel approach to finding the latter two efficiently.

D. Synthesis of Open-Loop Controller

In this section, we will discuss the synthesis of the open-loop controller $\mathbf{u}_{\text{ref}}$ and $\mathbf{x}_{\text{ref}}[0]$ in $\langle \mathbf{K}, \mathbf{x}_{\text{ref}}[0], \mathbf{u}_{\text{ref}} \rangle$. From the previous section, we know that given an initial set S, a tracking controller $\mathbf{K}$, and an open-loop controller $\mathbf{u}_{\text{ref}}$, the reachable set (under any disturbance) at time t is overapproximated by $E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t])$. Thus, once we fix $\mathbf{K}$, the problem of synthesizing a controller reduces to the problem of synthesizing an appropriate $\mathbf{u}_{\text{ref}}$ and $\mathbf{x}_{\text{ref}}[0]$ such that the reachset overapproximations meet the reach-avoid specification. Indeed, for the rest of the this section, we will assume fixed $\mathbf{K}$.

For synthesizing $\mathbf{u}_{\text{ref}}$ and $\mathbf{x}_{\text{ref}}[0]$, we would like to formalize the problem in terms of constraints that will allow us to use SMT solvers. As we have discussed in Section IV-A, the quantifier-free formulas are simpler than formulas with quantifier alternations [73]. In the following, we describe the details of how this problem can be formalized as a quantifier-free first-order formula over the theory of reals. We will then lay out specific assumptions and/or simplifications required to reduce the problem to `QF-LRA` theory, which is implemented effectively in existing state-of-the-art SMT solvers. Most SMT solvers also provide the functionality of explicit model generation, and the concrete controller values can be read-off from the models generated when the constraints are satisfiable.

1) Constraints for Synthesizing $\mathbf{u}_{\text{ref}}$: The $\mathbf{u}_{\text{ref}}$ synthesis problem can be stated as finding satisfying solutions for the formula ϕ_{synth}, where the initial set of states is $S = B_{\mathbf{r}[0]}(\mathbf{x}_{\text{ref}}[0])$

$$\begin{aligned}\phi_{\text{synth}} \triangleq\ & \exists \mathbf{u}_{\text{ref}}[0], \mathbf{u}_{\text{ref}}[1], \ldots \mathbf{u}_{\text{ref}}[T-1], \mathbf{r}[0] \\ & \exists \mathbf{x}_{\text{ref}}[0], \mathbf{x}_{\text{ref}}[1], \ldots \mathbf{x}_{\text{ref}}[T] \\ & \phi_{\text{control}}(\mathbf{u}_{\text{ref}}) \wedge \phi_{\text{execution}}(\mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}}) \\ & \wedge \phi_{\text{avoid}}(\mathbf{r}[0], \mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}}) \wedge \phi_{\text{reach}}(\mathbf{r}[0], \mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}})\end{aligned} \tag{13}$$

where ϕ_{control} constrains the space of inputs, $\phi_{\text{execution}}$ states that the sequence $\mathbf{x}_{\text{ref}}$ is a reference execution following (4), ϕ_{avoid} specifies the safety constraint, and ϕ_{reach} specifies that the system reaches G:

$$\begin{aligned}&\phi_{\text{control}}(\mathbf{u}_{\text{ref}}) \triangleq \bigwedge_{t=0}^{T-1} \mathbf{u}_{\text{ref}}[t] \oplus \big(\mathbf{K}[t] \otimes E_{\mathbf{r}[t]}(0, \mathbf{M}[t])\big) \subseteq U \\ &\phi_{\text{execution}}(\mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}}) \triangleq \\ &\qquad \bigwedge_{t=0}^{T-1} (\mathbf{x}_{\text{ref}}[t+1] = \mathbf{A}[t]\mathbf{x}_{\text{ref}}[t] + \mathbf{B}[t]\mathbf{u}_{\text{ref}}[t]) \\ &\phi_{\text{avoid}}(\mathbf{r}[0], \mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}}) \triangleq \bigwedge_{t=0}^{T} E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t]) \cap \mathbf{O}[t] = \emptyset \\ &\phi_{\text{reach}}(\mathbf{r}[0], \mathbf{u}_{\text{ref}}, \mathbf{x}_{\text{ref}}) \triangleq E_{\mathbf{r}[T]}(\mathbf{x}_{\text{ref}}[T], \mathbf{M}[T]) \subseteq G.\end{aligned} \tag{14}$$

We make a few remarks about this formulation. First, each of the formulas ϕ_{control}, ϕ_{avoid}, and ϕ_{reach} represent sufficient conditions to check for the existence of $\mathbf{u}_{\text{ref}}$. Second, the constraints stated previously belong to the (decidable) theory of reals. However, ϕ_{control}, ϕ_{avoid}, and ϕ_{reach}, and thus, ϕ_{synth}, are not quantifier free as they use subset and disjointness checks. This is because for sets S, T expressed as predicates $\varphi_S(\cdot)$ and $\varphi_T(\cdot)$, $S \cap T = \emptyset$ corresponds to the formula $\forall x \cdot \neg(\varphi_S(x) \wedge \varphi_T(x))$ and $S \subseteq T$ (or equivalently $S \cap T^c = \emptyset$) corresponds to the formula $\forall x \cdot \varphi_S(x) \Rightarrow \varphi_T(x)$.

2) Reduction to `QF-LRA`: The central idea behind eliminating the universal quantification in the disjointness predicates in ϕ_{avoid}, or in the inferred disjointness predicates in ϕ_{reach} and ϕ_{control}, is to check whether an ellipsoid is disjointed or contained in a polytope. Lemmas 2 and 3 state that the disjointness and containment checks can be done through linear constraints.

Lemma 2: For an ellipsoid $E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t])$ and a polytope $\{x \in \mathbb{R}^k \mid Ax \leq b\}$, if

$$\begin{aligned}&\bigvee_{i=1}^{k} \Big(A^{(i)}\mathbf{x}_{\text{ref}}[t] > b(i)\Big) \wedge \\ &\left(\frac{A^{(i)}\mathbf{x}_{\text{ref}}[t] - b(i)}{\|\tilde{A}^{(i)}\|_2} > \mathbf{r}[t] \vee \frac{A^{(i)}\mathbf{x}_{\text{ref}}[t] - b(i)}{\|\tilde{A}^{(i)}\|_2} < -\mathbf{r}[t]\right)\end{aligned} \tag{15}$$

where $\tilde{A} = A\mathbf{M}^{-1}[t]$, then

$$E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t]) \cap \{x \mid Ax \leq b\} = \emptyset.$$

Proof: Take an affine coordinate transformation $y = \mathbf{M}[t]x$ and let $\tilde{\mathbf{x}}_{\text{ref}}[t] = \mathbf{M}[t]\mathbf{x}_{\text{ref}}[t]$. Under the transformed coordinate, the ellipsoid $E_{\mathbf{r}[t]}(\mathbf{x}_{\text{ref}}[t], \mathbf{M}[t])$ becomes a ball

$$E_{\mathbf{r}[t]}(\mathbf{M}[t]\mathbf{x}_{\text{ref}}[t], I) = B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\text{ref}}[t])$$

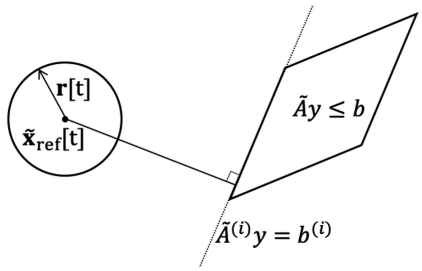

Fig. 2. Illustration of $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ being disjointed from the polytope $\{y \mid \tilde{A}y \leq b\}$.

and the polytope also becomes $\tilde{A}y \leq b$. Affine transformation preserves the disjointness between objects. As long as the ball $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ is disjointed from the polytope $\tilde{A}y \leq b$, the original ellipsoid and polytope are disjointed.

Consider the ball $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ in the transformed coordinate, if the center $\tilde{\mathbf{x}}_{\mathsf{ref}}[t]$ is outside the polytope $\tilde{A}y \leq b$ and its distance to an surface of the polytope is greater than $\mathbf{r}[t]$, then the ball $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ is not intersecting with any surfaces of the polytope, and therefore, is disjointed from the polytope (as shown in Fig. 2). Equivalently, this means that there exists an $i \leq k$, such that $\tilde{A}^{(i)}\tilde{\mathbf{x}}_{\mathsf{ref}}[t] > b(i)$, and the distance from $\tilde{\mathbf{x}}_{\mathsf{ref}}[t]$ to any surface, which is a hyperplane $\tilde{A}^{(i)}x = b(i)$, is greater than $\mathbf{r}[t]$. Recall that $A^{(i)}$ and $b(i)$ are the ith row of A and b, respectively.

The distance from $\tilde{\mathbf{x}}_{\mathsf{ref}}[t]$ to a hyperplane $\tilde{A}^{(i)}x = b(i)$ is $\frac{|\tilde{A}^{(i)}\tilde{\mathbf{x}}_{\mathsf{ref}}[t]-b(i)|}{\|\tilde{A}^{(i)}\|_2}$. Therefore, the ball $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ is disjointed from the polytope $\{y \mid \tilde{A}y \leq b\}$ if the following is true:

$$\bigvee_{i=1}^{k}\left(\tilde{A}^{(i)}\tilde{\mathbf{x}}_{\mathsf{ref}}[t] > b(i)\right) \wedge \left(\frac{\tilde{A}^{(i)}\tilde{\mathbf{x}}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} > \mathbf{r}[t] \vee \frac{\tilde{A}^{(i)}\tilde{\mathbf{x}}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} < -\mathbf{r}[t]\right)$$

which is equivalent to (15). ■

In Lemma 2, to check whether an ellipsoid is disjointed from a polytope (obstacle) with k surfaces using (15), the formula contains 3 k linear inequalities with conjunctions and disjunctions. In [26], the reach set overapproximations are represented using hyper-rectangles. The hyper-rectangle is disjointed from the polytope if there is a surface of the polytope such that the vertices of the hyper-rectangle lie on the other side of the surface. Such a formula has $2^n k$ linear inequalities, where n is the dimensionality of the state space. Compared with the methods used in [26], Lemma 2 reduces the number of constraints in ϕ_{avoid} from $2^n k$ to 3 k, which is the key fact that makes the proposed approach scale to systems with large n. We will also see the same improvement in ϕ_{reach} and ϕ_{control}.

Similar to Lemma 2, as long as the center of the ball $B_{\mathbf{r}[t]}(\tilde{\mathbf{x}}_{\mathsf{ref}}[t])$ is inside the polytope $\tilde{A}y \leq b$, and the distances from $\tilde{\mathbf{x}}_{\mathsf{ref}}[t]$ to all surfaces of the polytope $\tilde{A}^{(i)}x = b(i)$ are greater than the radius $\mathbf{r}[t]$, the ball is entirely contained in the polytope.

Lemma 3: For any ellipsoid $E_{\mathbf{r}[t]}(\mathbf{x}_{\mathsf{ref}}[t], \mathbf{M}[t])$ and a polytope $\{x \in \mathbb{R}^k \mid Ax \leq b\}$, if

$$\bigwedge_{i=1}^{k}\left(A^{(i)}\mathbf{x}_{\mathsf{ref}}[t] \leq b(i)\right) \wedge \left(\frac{A^{(i)}\mathbf{x}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} \geq \mathbf{r}[t] \vee \frac{A^{(i)}\mathbf{x}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} \leq -\mathbf{r}[t]\right) \tag{16}$$

where $\tilde{A} = AM^{-1}[t]$, then

$$E_{\mathbf{r}[t]}(\mathbf{x}_{\mathsf{ref}}[t], \mathbf{M}[t]) \subseteq \{x \mid Ax \leq b\}.$$

With Lemma 2 and 3, we can rewrite ϕ_{avoid} and ϕ_{reach} in (14) as

$$\begin{aligned}\phi_{\mathsf{avoid}}(\mathbf{r}[0], \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) \triangleq \bigwedge_{t=0}^{T}\bigwedge_{\{x|Ax\leq b\}\in\mathbf{O}[t]}\bigvee_{i=1}^{k}\left(A^{(i)}\mathbf{x}_{\mathsf{ref}}[t] > b(i)\right) \\ \wedge \left(\frac{A^{(i)}\mathbf{x}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} > \mathbf{r}[t] \vee \frac{A^{(i)}\mathbf{x}_{\mathsf{ref}}[t]-b(i)}{\|\tilde{A}^{(i)}\|_2} < -\mathbf{r}[t]\right)\end{aligned}$$

$$\begin{aligned}\phi_{\mathsf{reach}}(\mathbf{r}[0], \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) \triangleq \bigwedge_{i=1}^{k}\left(A_G^{(i)}\mathbf{x}_{\mathsf{ref}}[T] \leq b_G(i)\right) \\ \wedge \left(\frac{A_G{}^{(i)}\mathbf{x}_{\mathsf{ref}}[T]-b_G(i)}{\|\tilde{A_G}{}^{(i)}\|_2} \geq \mathbf{r}[T] \vee \frac{A_G{}^{(i)}\mathbf{x}_{\mathsf{ref}}[T]-b_G(i)}{\|\tilde{A_G}{}^{(i)}\|_2} \leq -\mathbf{r}[T]\right)\end{aligned} \tag{17}$$

where in ϕ_{reach}, the goal set G is represented as an ellipsoid $\{x|A_G x \leq b_G\}$. Once the tracking controller $\mathbf{K}$ is fixed, the matrices $\tilde{A}$ (or $\tilde{A_G}$) are constants. Moreover, $\mathbf{r}[t] = \mathbf{r}[0] + \sum_{i=0}^{t-1}\delta[i]$ and δ are also constants. Therefore, ϕ_{avoid} and ϕ_{reach} are linear expressions of $\mathbf{r}[0], \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}$ with disjunctions. In the expression ϕ_{control} of (14), $\mathbf{u}_{\mathsf{ref}}[t] \oplus (\mathbf{K}[t] \otimes E_{\mathbf{r}[t]}(0, \mathbf{M}[t]))$ is essentially also an ellipsoid $E_{\mathbf{r}[t]}(\mathbf{u}_{\mathsf{ref}}[t], \mathbf{M}[t]\mathbf{K}^{-1}[t])$. Therefore, ϕ_{control} can also be represented as a linear expression of $\mathbf{u}_{\mathsf{ref}}$ and $\mathbf{r}[0]$.

As discussed previously, the constraints as in $\phi_{\mathsf{control}}, \phi_{\mathsf{execution}}$, ϕ_{avoid}, and ϕ_{reach} only give rise to linear constraints, do not have the $\forall$ quantification over states, and are sound transformations of ϕ_{synth} into `QF_LRA`. Moreover, the number of linear inequality constraints in ϕ_{synth} is only $O(kT)$, where T is the number of time steps T, and k is the number of surfaces in obstacles and the goal set. In Section IV-E, we will see that as the reach sets are exact when the disturbance is 0 (see Corollary 1), these checks will also turn out to be sufficient to ensure that if there exists a controller, ϕ_{synth} is satisfiable.

Lemma 4: If the formula ϕ_{synth} is satisfiable, then there is a control sequence $\mathbf{u}_{\mathsf{ref}}$ such that for every $x \in B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0])$ and for every $\mathbf{d} \in D^T$, the unique execution $\mathbf{x}$ defined by the controller $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}}\rangle$ and $\mathbf{d}$, starting at x, satisfies $\mathbf{x}[T] \in G \wedge \forall t \leq T \cdot \mathbf{x}[t] \notin \mathbf{O}[t]$.

We remark that a possible alternative for eliminating the $\forall$ quantifier is the use of Farkas' lemma, but this gives rise to

nonlinear constraints.[1] Indeed, in our experimental evaluation, we observed the downside of resorting to Farkas' lemma in this problem.

We also remark that the SAT encoding as in Lemma 2 can be formulated as mixed integer linear constraints using the "big-M" method to get rid of the disjunction operators $\vee$, by introducing extra auxiliary integer variables (see details in [75]). Then, ϕ_{synth} in (13) can be solved through solving a MILP or MIQP problem. In this way, our encoding for $\phi_{\mathsf{control}}, \phi_{\mathsf{execution}}$, and ϕ_{avoid} (as mixed integer linear formulae $\phi'_{\mathsf{control}}, \phi'_{\mathsf{execution}}$, and ϕ'_{avoid} using the "big-M" method on the original formulae) can be used in dynamic and real-time control using MPC, where the obstacles $\mathbf{O}[t]$ are constructed dynamically as system evolves, and Θ is a set instead of a single point due to bounded localization errors

$$\begin{aligned} \min_{\substack{\mathbf{u}_{\mathsf{ref}}[0],\ldots\mathbf{u}_{\mathsf{ref}}[N-1],\\ \mathbf{r}[0],\mathbf{x}_{\mathsf{ref}}[0],\ldots\mathbf{x}_{\mathsf{ref}}[N]}} \quad & V_f(\mathbf{x}_{\mathsf{ref}}[N]) + \sum_{i=0}^{N-1} \ell(\mathbf{x}_{\mathsf{ref}}[i], \mathbf{u}_{\mathsf{ref}}[i]) \\ \text{s.t.} \quad & \phi'_{\mathsf{control}}(\mathbf{u}_{\mathsf{ref}}) \wedge \phi'_{\mathsf{execution}}(\mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) \\ & \wedge \phi'_{\mathsf{avoid}}(\mathbf{r}[0], \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) \\ & \wedge (\mathbf{x}_{\mathsf{ref}}[0] = \mathsf{center}(\Theta)) \\ & \wedge (\mathbf{r}[0] \geq \mathsf{diameter}(\Theta)) \,. \end{aligned} \tag{18}$$

We implemented both the SAT encoding as in (17) and the corresponding mixed integer linear encoding with the objective function $\|x_{\mathsf{ref}}[N] - \mathsf{center}(G)\|_2$ (using the Gurobi solver), and observe that both the two encoding has no major difference in terms of running time when $N = T$. Moreover, both SAT over `QF_LRA` and MIP problems are NP-hard [76].

E. Synthesis Algorithm Putting it All Together

Section IV-D describes how to formalize constraints to generate a control sequence that works for S, which could be a subset of the initial set Θ. The overall synthesis procedure (see Algorithm 1), first computes a tracking controller $\mathbf{K}$, then generates open-loop control sequences and reference executions in order to cover the entire set Θ.

The procedure ReachParams computes the tracking controller $\mathbf{K}$, based on which it further computes a sequence of shape matrices $\mathbf{M}$ and disturbance bounds δ using Lemma 1, for the system $\mathcal{A}$ and time bound T with $\mathbf{Q}, \mathbf{R}$ for the LQR method. Given any reference execution $\mathbf{x}_{\mathsf{ref}}$ and initial set $B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0])$, the parameters computed by ReachParams can be used to overapproximate $\mathsf{Reach}(B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0]), t)$ with the ellipsoid $E_{\mathbf{r}[t]}(\mathbf{x}_{\mathsf{ref}}[t], \mathbf{M}[t])$, where $\mathbf{r}[t] = \mathbf{r}[0] + \sum_{i=0}^{t-1} \delta[i]$.

The procedure getConstraints constructs the logical formula ψ_{synth} such that whenever ψ_{synth} holds, we can find an initial radius $\mathbf{r}[0]$ that is abovementioned some threshold t^*, and center $\mathbf{x}_{\mathsf{ref}}[0]$ in the set $\Theta \setminus \mathtt{cover}$ and a control sequence $\mathbf{u}_{\mathsf{ref}}$ such that any controlled execution starting from $B_{\mathbf{r}}[0](\mathbf{x}_{\mathsf{ref}}[0])$ satisfies the reach-avoid requirements

$$\psi_{\mathsf{synth}} \triangleq \phi_{\mathsf{synth}} \wedge \mathbf{x}_{\mathsf{ref}}[0] \in \Theta \wedge \mathbf{x}_{\mathsf{ref}}[0] \notin \mathtt{cover} \wedge \mathbf{r}[0] > r^*. \tag{19}$$

[1] Farkas' lemma introduces auxiliary variables that get multiplied with existing variables $\mathbf{x}_{\mathsf{ref}}[0], \ldots, \mathbf{x}_{\mathsf{ref}}[T]$, leading to nonlinear constraints.

Algorithm 1: Algorithm for Synthesizing Combined Controller.

```
input  : A, T, O, G, Q, R
output : controllers = {(⟨K, x_ref[0], u_ref⟩, B_r[0](x_ref[0]))}
initially: r* ← diameter(Θ)/2 ;
K, M, δ ← ReachParams(A, T, Q, R) ;
cover ← ∅;
controllers ← ∅
1 while Θ ⊈ cover do
2     ψ_synth ← getConstraints(A, T, O, G, M, δ, r*, cover) ;
3     if CHECKSAT(ψ_synth) = SAT then
4         r[0], u_ref, x_ref ← model(ψ_synth) ;
5         cover ← cover ∪ B_r[0](x_ref[0]);
6         controllers ← controllers ∪
            { ( ⟨K, x_ref[0], u_ref⟩ , B_r[0](x_ref[0]) ) } ;
7     else
8         r* ← r*/2 ;
9 return controllers ;
```

Line 3 checks for the satisfiability of ψ_{synth}. If satisfiable, we extract the model generated to get the radius of the initial ball, the control sequence $\mathbf{u}_{\mathsf{ref}}$, and the reference execution $\mathbf{x}_{\mathsf{ref}}$ in Line 4. The generated controller $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ is guaranteed to work for the ball $B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0])$, which can be marked *covered* by adding it to the set `cover`. In order to keep all the constraints linear, one can further underapproximate $B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0])$ with a hypercube $\{x \in \mathbb{R}^n \mid \wedge_{i=1}^n \mathbf{x}_{\mathsf{ref}}[0](i) - \mathbf{r}[0](i)/\sqrt{n} \leq x \leq \mathbf{x}_{\mathsf{ref}}[0](i) + \mathbf{r}[0](i)/\sqrt{n}\}$. If ψ_{synth} is unsatisfiable, then we reduce the minimum radius r^* (see Line 8) and continue to look for controllers, until we find that $\Theta \subseteq \mathtt{cover}$.

The set `controllers` is the set of pairs $(\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle, S)$, such that the controller $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ drives the set S to meet the desired specification. Each time a new controller is found, it is added to the set `controllers` together with the initial set for which it works (see Line 6).

The following theorem asserts the soundness of Algorithm 1, and it follows from Lemmas 1 and 4.

Theorem 1: If Algorithm 1 terminates, then the synthesized controller is correct. That is, (a) for each $x \in \Theta$, there is a $(\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle, S) \in \mathtt{controllers}$, such that $x \in S$, and (b) for each $(\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle, S) \in \mathtt{controllers}$, the unique controller $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ is such that for every $x \in S$ and for every $\mathbf{d} \in D^T$, the unique execution defined by $\langle \mathbf{K}, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ and $\mathbf{d}$ [as in (2) and (4)], starting at x, satisfies the reach-avoid specification.

Algorithm 1 ensures that, upon termination, every $x \in \Theta$ is covered, i.e., one can construct a combined controller that drives x to G while avoiding $\mathbf{O}$. However, it may find multiple controllers for a point $x \in \Theta$. This nondeterminism can be easily resolved by picking any controller assigned for x.

Below, we show that, under certain robustness assumptions on the system $\mathcal{A}$, G and the sets $\mathbf{O}$, and in the absence of disturbance, Algorithm 1 terminates.

Robustly controllable systems: A system $\mathcal{A} = \langle \mathbf{A}, \mathbf{B}, \Theta, U, D \rangle$ is said to be ε-robustly controllable ($\varepsilon > 0$) with respect to the reach-avoid specification ($\mathbf{O}$, G) and matrices $\mathbf{K}$, if (a) $D = \{0\}$, and (b) for every initial state $\theta \in \Theta$ there is an open loop-controller $\mathbf{u}_{\mathsf{ref}} \in U^T$ such that the unique execution starting from θ using the open-loop controller $\mathbf{u}_{\mathsf{ref}}$ satisfies the reach-avoid specification. Moreover, with the controller $\langle \mathbf{K}, \theta, \mathbf{u}_{\mathsf{ref}} \rangle$ defined as in (4), $\forall x \in B_\varepsilon(\theta)$, the unique trajectory $\mathbf{x}$ defined by the controller $\langle \mathbf{K}, \theta, \mathbf{u}_{\mathsf{ref}} \rangle$ starting from x also satisfies the reach avoid specification.

Theorem 2: If $\mathcal{A}$ is an ε-robust controllable system with respect to the reach-avoid specification $(\mathbf{O}, G)$, the tracking controller $\mathbf{K}$, and an arbitrarily small $\varepsilon > 0$, then Algorithm 1 terminates.

Proof: As seen in Corollary 1, when the system is robust, then (in the absence of any disturbance, i.e., $D = \{0\}$), the computed ellipsoids are exact reach sets starting from $B_{\mathbf{r}[0]}(\mathbf{x}_{\mathsf{ref}}[0])$. Moreover, as r^* approaches 0, $\mathbf{r}[0]$ can also approach 0. From Corollary 1, we know that $\forall t \geq 0, \mathbf{r}[t] = \mathbf{r}[0]$, so the radii of the reach sets ellipsoids all converge to 0. With $\mathbf{r}[t] \to 0$, (15) and (16) in Lemmas 2 and 3 [therefore, (17)] also become satisfiable whenever there is a controller. The correctness of Theorem 2 then follows from the above observations. ■

We remark that an alternative approach to solve the bounded controller synthesis problem is to synthesize an open-loop control sequence $\mathbf{u}_{\mathsf{ref}}$ for a single initial condition $\mathbf{x}_{\mathsf{ref}}[0]$ first, and then find the maximum cover such that there exists a tracking controller $\mathbf{K}$ to make every execution starting from the cover also satisfy the reach-avoid specification. However, when implemented this approach, we observed that the synthesized reference trajectory $\mathbf{x}_{\mathsf{ref}}$ always got very close to the obstacles. Therefore, the maximum initial cover for which this reference trajectory works would be minuscule, and result in a very large number of partitions in the initial set. In contrast, Algorithm 1 asks the SMT solver to search for a reference that works for an initial cover with the size of at least r^* with any disturbance (and r^* is adjusted iteratively), resulting in a much smaller solution space.

V. RealSyn ver 2.0 Implementation and Evaluation

For experimental evaluation, we have implemented Algorithm 1, the tool RealSyn ver 2.0. The previous version of the tool, RealSyn ver 1.0, appeared in our earlier paper [26]. The key distinction in the new implementation is the encoding of the reach-avoid constraints, as in Lemmas 2 and 3. As a result, the final formulas for the reach-avoid constraints (17) for synthesizing the open-loop controller consist of $O(k)$ linear constraints, with k being the number of hyperplanes of the obstacles and the goal set. In contrast, in RealSyn ver 1.0, such formulas have $O(2^n k)$ linear constraints, where n is the dimensionality of the state space.

For solving (19), RealSyn ver 2.0 can use any SMT solver as a subroutine. For our results here we use `Yices` [77], as it outperformed the other solvers in [26]. We evaluate our approach on 10 example synthesis problems (from [26]) on a standard laptop with Intel Core i7 processor, 16 GB RAM. The results are reported in Table I. Overall, our results demonstrate the effectiveness of using our approach and the feasibility of scalable controller synthesis for high-dimensional systems and complex reach-avoid specifications.

Comparison with other tools: We considered other controller synthesis tools for possible comparison with RealSyn ver 2.0. In brief, CoSyMa [56], Pessoa [4], and SCOTS [36] do not explicitly support discrete-time systems. LTLMop [57], [58] is designed to analyze models in the 2-D Euclidean plane, and therefore, is not suitable for most of our examples. TuLiP [44], [59] comes closest to addressing the same class of problems. TuLip relies on discretization of the state space and a receding horizon approach for synthesizing controllers for more general GR(1) specifications. However, we found that TuLip succumbs to the state-space explosion problem when discretizing the state space, and it did not work on most of our examples. For instance, TuLiP was unable to synthesize a controller for the 2-D system "1-robot" (see Table I), and returned `unrealizable`. On the benchmark "2-robot" ($n = 4$), TuLip did not return any answer within 1 h. SMC [27], [28], as discussed in Section II, is the closest to ours as in solving reach-avoid problems, and the only one among the tools that can return comparable results to RealSyn ver 2.0. We adopt the implementation of SMC as in[2] to be used on our benchmarks and report results in Table I.

Benchmarks: Our benchmarks are mainly vehicle motion planning problems with reach-avoid specifications. Benchmarks 1 and 2 model robots moving on the Euclidean plane, where each robot is a 2-D system and admits a 1-D input. Starting from some initial region on the plane, the robots are required to reach the common goal area within the given time steps, while avoiding certain obstacles. For "2-robot," the robots are also required to maintain a minimum separation. Benchmarks 3–7 are discrete vehicular models adopted from [70]. Each vehicle is a 4-D linear system with 2-D input. Benchmark 3 from [26] describes a mobile robot needs to accomplish a reach-avoid goal in an apartment. Benchmark 4 describes a vehicle running on a two-lane road, trying to overtake a vehicle in front of it. The second vehicle serves as the dynamic obstacle. Benchmarks 5–7 are similar to Benchmark 2, where the vehicles are required to reach a common goal area while avoiding collision with the obstacles and with each other (inspired by a merge). The velocities and accelerations of the vehicles are also constrained in each of these benchmarks. Fig. 3 shows the setting for three vehicles trying to reach the green goal set while avoiding the red obstacle and maintaining a distance of > 0.5 (m) all the time. Fig. 3 also shows the reachsets of each vehicle projected to the 2-D plane of vehicles' positions. We observe from Fig. 3 that to make sure that all vehicles do not collide with each other, the synthesized controller forces the vehicles to arrive at the goal set at different time steps.

Benchmarks 8–10 model multiple vehicles trying to form a platoon by maintaining the safe relative distance between consecutive vehicles. The models are adopted (and discretized) from [19]. Each vehicle is a 2-D system with 1-D input. For the

[2]"SMC-LTL Github Repository," [Online]. Available: https://github.com/rcpsl/SMC-LTL

TABLE I
RUN TIME PERFORMANCE COMPARISON OF CONTROLLER SYNTHESIS USING REALSYN VER 2.0 WITH THE ORIGINAL SYNTHESIS ALGORITHM REALSYN VER 1.0 AS IN [26] AND SMC [27]

	Model	n	m	Algorithm 1		CAV Algorithm [26]		SMC [27]	
				#iter	time(s)	#iter	time(s)	#iter	time(s)
1	1-robot	2	1	7	0.03	7	0.06	1	0.09
2	2-robot	4	2	1	0.04	183	2.26	1	1.69
3	running-example in [26]	4	2	1	104	1	319.97	1	227.86
4	1-car dynamic avoid	4	2	12	8.12	12	8.49	1	15.58
5	1-car navigation	4	2	15	1.14	17	6.73	1	5.37
6	2-car navigation	8	4	1	1.86	1	4.07	1	6.07
7	3-car navigation	12	6	1	4.70	1	741.73	1	372.48
8	4-car platoon	8	4	1	0.03	1	0.15	1	0.33
9	8-car platoon	16	8	1	0.10	1	0.62	1	0.94
10	10-car platoon	20	10	1	0.12	1	7.74	1	5.72

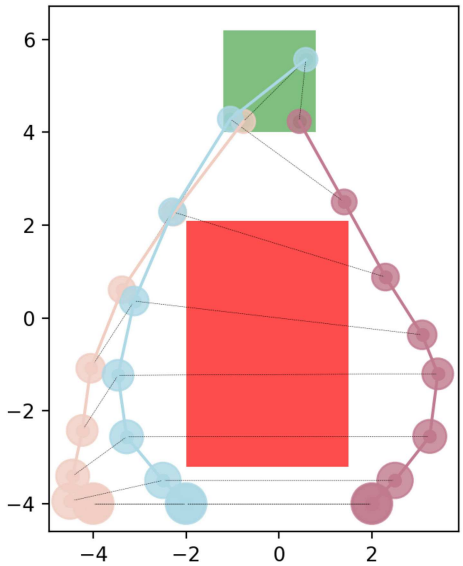

Fig. 3. Reachsets of three cars with synthesized controller for reach-avoid specification. Ellipsoids represent the projection of the reachset on the vehicle's position on the 2-D plane. Ellipsoids of the same color connected by the line of same color belong to the same vehicle. Reachsets of the same time step are connected using the black dotted line. Red polytope is the obstacle and green polytope is the goal set. Note that different vehicles arrive at the goal set at different time steps so they do not collide with each other, although some ellipsoids (at different time steps) appear to overlap.

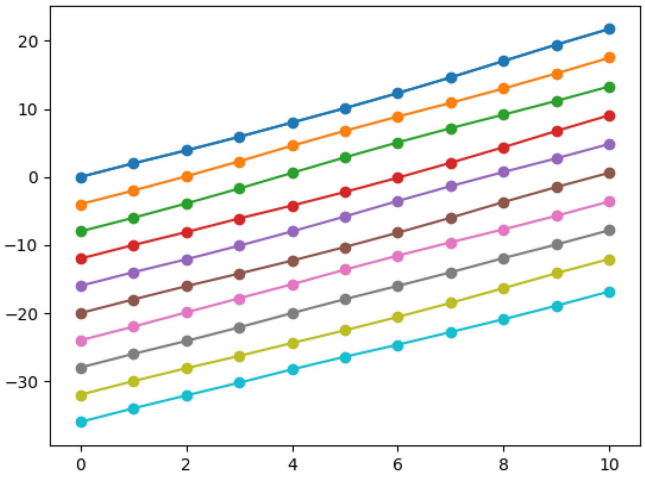

Fig. 4. Ten cars are forming a platoon with synthesized controller. The x-axis is time and the y-axis is the position of each car.

four-car platoon model, the running times reported in Table I are much smaller than the time (5 min) reported in [19]. This observation aligns with our analysis in Section IV-A. For the 10-car platoon case, Fig. 4 shows the positions of the cars along time with the synthesized controller using REALSYN VER 2.0. As shown in Fig. 4, all vehicles are maintaining a safe relative distance $>$ 1(m) to its neighbor vehicles even with disturbances.

Synthesis performance: In Table I, columns "n" and "m" stand for the dimensions of the state space and input space. For each background solver, "#iter" is the number of iterations Algorithm 1 required to synthesize a controller, and "time" is the respective running times. All benchmarks are synthesized for a specification with 10–20 steps.

In general, the proposed algorithm improves the performance of REALSYN VER 2.0 with the running time 2–150 times faster than REALSYN VER 1.0 as in [26], and 2–80 times faster than SMC as in. The only exception is Benchmark 4 where the running time stays almost the same for REALSYN VER 1.0 and REALSYN VER 2.0. This is because in Benchmark 4, all obstacles, goal set, and reach set overapproximations in [26] were represented as axis-aligned hyper-rectangles. To check the disjointness and containment of axis-aligned hyper-rectangles, [26] used a much simpler method with $O(n)$ linear inequalities, instead of enumerating all the vertices of the hyper-rectangles, which introduces $O(2^n)$ linear inequalities. Therefore, the improvement of the proposed algorithm in this paper on Benchmark 4 is minor over REALSYN VER 1.0.

However, for the rest of the benchmarks, where the obstacles are not axis-aligned hyper-rectangles, the proposed new algorithm can reduce the number of linear constraints in the final SAT problem [(19)] from $O(2^n)$ to $O(1)$ with respect to the dimensionality of the system, comparing with REALSYN VER 1.0. The results in Table I substantiate our analysis in Section IV. SMC needs to discretize the freespace (complimentary of the obstacles) into convex regions for motion planning problems.

Therefore, the complexity of the SMC problem relies on the number of convex regions. We observe from Table I one that SMC performs comparable to our proposed method for lower dimensional problems and much slower on higher dimensional examples.

VI. Conclusion

In this article, we proposed a novel technique for synthesizing controllers for systems with time-varying discrete-time linear dynamics, operating under bounded disturbances, and for reach-avoid specifications. Our approach relies on generating controllers that combine an open-loop controller with a tracking controller, thereby allowing a decoupled approach for synthesizing each component independently. Experimental evaluation using our tool RealSyn ver 2.0 demonstrates the value of the approach when analyzing systems with complex dynamics and specifications.

Acknowledgment

The authors acknowledge supports from DARPA, NSF, and AFSOR. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This work was done while Q. Ning was with the Allen Institute for AI.

References

[1] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proc. 44th IEEE Conf. Decis. Control*, 2005, pp. 4885–4890.

[2] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, Feb. 2009.

[3] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, no. vol. 1, pp. 211–236, 2018.

[4] P. Roy, P. Tabuada, and R. Majumdar, "Pessoa 2.0: A controller synthesis tool for cyber-physical systems," in *Proc. 14th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, ACM, Apr. 2011, pp. 315–316.

[5] J. P. Hespanha, *Linear Systems Theory*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[6] P. J. Antsaklis and A. N. Michel, A Linear Systems Primer, vol. 1. Boston, MA, USA: *Birkhäuser*, 2007.

[7] J. Ding and C. J. Tomlin, "Robust reach-avoid controller synthesis for switched nonlinear systems," in *Proc. 49th IEEE Conf. Decis. Control*,, Atlanta, GA, USA, 2010, pp. 6481–6486.

[8] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri, "Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm," in *Proc. 54th IEEE Conf. Decis. Control*, Osaka, Japan, 2015, pp. 7434–7439.

[9] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Control*, Seattle, WA, USA, 2015, pp. 11–20.

[10] P. M. Esfahani, D. Chatterjee, and J. Lygeros, "The stochastic reach-avoid problem and set characterization for diffusions," *Automatica*, vol. 70, pp. 43–56, Aug. 2016.

[11] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: A. modular framework for fast and guaranteed safe motion planning," in *Proc. IEEE 56th Annu. Conf. Decis. Control*,, Dec.2018, pp. 1517–1522.

[12] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *Int. J. Robot. Res.*, vol. 39, no. 12, pp. 1419–1469, Sep. 2018.

[13] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, and C. Tomlin, "Tunnel-milp: Path planning with sequential convex polytopes," in *Proc. AIAA Guid., Navigation Control Conf. Exhibit*, 2008, Art. no. 7132.

[14] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Control*, Apr. 2015, pp. 239–248.

[15] J. M. Filho, E. Lucet, and D. Filliat, "Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 657–663.

[16] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 174–179.

[17] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Berlin, Germany: Springer Science and Business Media, Jun. 2009.

[18] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*, vol. 89. Berlin, Germany: Springer, Jan. 2017.

[19] B. Schürmann and M. Althoff, "Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems," in *Proc. Amer. Control Conf.*, Seattle, WA, USA, May 2017, pp. 2522–2529.

[20] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[21] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal techniques for reachability analysis of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 52, no. 1, pp. 26–38, Jan. 2007.

[22] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[23] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas, "Robust test generation and coverage for hybrid systems," in *Proc. Int. Workshop Hybrid Syst., Comput. Control*, Berlin, Germany, Springer, pp. 329–342, Apr. 2007.

[24] C. Fan, J. Kapinski, X. Jin, and S. Mitra, "Locally optimal reach set over-approximation for nonlinear systems," in *Proc. Int. Conf. Embedded Softw.*, 2016, pp. 1–10.

[25] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, Oct. 2008.

[26] C. Fan, U. Mathur, S. Mitra, and M. Viswanathan, "Controller synthesis made real: Reachavoid specifications and linear dynamics," in *Computer Aided Verification.*New York, NY, USA: Springer, 2018, pp. 347–366.

[27] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex optimization," in *Proc. 20th Int. Conf. Hybrid Syst., Comput. Control*, 2017, pp. 19–28.

[28] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex programming," *Proc. IEEE*, vol. 106, no. 9, pp. 1655–1679, Sep. 2018.

[29] M. A. Rami and F. Tadeo, "Controller synthesis for positive linear systems with bounded controls," *IEEE Trans. Circuits Syst.*, vol. 54-II, no. no. 2, pp. 151–155, Feb. 2007.

[30] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb. 2008.

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

[32] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, Dec. 2006.

[33] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming - the explicit solution," *IEEE Trans. Autom. Control*, vol. 47, no. 12, pp. 1974–1985, 1974–1985, Dec. 2002.

[34] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1524–1534, Jul. 2011.

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

[36] M. Rungger and M. Zamani, "Scots: A tool for the synthesis of symbolic controllers," in *Proc. 19th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, ACM, Apr. 2016, pp. 99–104.

[37] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 58, no. 7, pp. 1771–1785, Jul. 2013.
[38] A. Abate, S. Amin, M. Prandini, J. Lygeros, and S. Sastry, "Computational approaches to reachability analysis of stochastic hybrid systems," in *Proc. Int. 10th Workshop Hybrid Syst., Comput. Control*, Pisa, Italy, Apr. 2007, pp. 4–17.
[39] R. W. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *IEEE Trans. Autom. Control*, vol. 45, no. 7, pp. 1279–1289, Jul. 2000.
[40] D. Liberzon, "Observer-based quantized output feedback control of nonlinear systems," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 8039–8043, 2008.
[41] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947–953, May 2012.
[42] A. Abate *et al.*, "Automated formal synthesis of digital controllers for state-space physical plants," in *Proc. 29th Int. Conf. Comput. Aided Verification*, Heidelberg, Germany, Jul. 2017, 462–482.
[43] R. Majumdar, K. Mallik, and A. Schmuck, "Compositional synthesis of finite state abstractions," *IEEE Trans. Autom. Control*, vol. 64, no. 6, pp. 2629–2636, Jun. 2019.
[44] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: A software toolbox for receding horizon temporal logic planning," in *Proc. 14th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, ACM, Apr. 2011, pp. 313–314.
[45] E. A. Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for linear systems," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1163–1176, May 2014.
[46] P. Bouyer, L. Bozzelli, and F. Chevalier, "Controller synthesis for mtl specifications," in *Proc. Int. Conf. Concurrency Theory*, Springer, Jan. 2006, pp. 450–464.
[47] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in*Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 81–87.
[48] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014.
[49] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.
[50] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Real-Time Trajectory Generation Using Model Predictive Control," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Aug. 2015, pp. 942–948.
[51] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, vol. 384. Berlin, Heidelberg: Springer, May 2009, pp. 345–369.
[52] J. A. Primbs, V. Nevistić, and J. C. Doyle, "Nonlinear optimal control: A control Lyapunov function and receding horizon perspective," *Asian J. Control*, vol. 1, no. 1, pp. 14–24, 1999.
[53] P. Ogren, M. Egerstedt, and X. Hu, "A control Lyapunov function approach to multi-agent coordination," in *Proc. 40th IEEE Conf. Decis. Control (Cat. no. 01CH37228)*, 2001, vol. 2, pp. 1150–1155.
[54] R. Freeman and P. V. Kokotovic, *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*. Berlin, Germany: Springer Science and Business Media, 2008.
[55] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.
[56] S. Mouelhi, A. Girard, and G. Gössler, "Cosyma: A tool for controller synthesis using multi-scale abstractions," in *Proc. 16th Int. Conf. Hybrid Syst., Comput. Control*, New York, NY, USA, ACM, Apr. 2013, pp. 83–88.
[57] K. W. Wong, C. Finucane, and H. Kress-Gazit, "Provably-correct robot control with ltlmop, OMPL and ROS," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, Nov. 2013, Art. no. 2073.
[58] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic based reactive mission and motion planning," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009.
[59] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray, "Control design for hybrid systems with tulip: The temporal logic planning toolbox," in *Proc. IEEE Conf. Control Appl.*, Buenos Aires, Argentina, Sep. 2016, pp. 1030–1041.
[60] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1781–1796, Apr. 2016.
[61] K. Hsu, R. Majumdar, K. Mallik, and A.-K. Schmuck, "Multi-layered abstraction-based controller synthesis for continuous-time systems," in *Proc. 21st Int. Conf. Hybrid Syst., Comput. Control (part CPS Week)*, Apr. 2018, pp. 120–129.
[62] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, vol. 1994. Stanford, CA, USA: Stanford University, 1994.
[63] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Autom. Symp. Proc. (Cat. no. 00CH37065)*, 2000, vol. 2, pp. 995–1001.
[64] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 883–921, Jun. 2015.
[65] H. Ravanbakhsh and S. Sankaranarayanan, "Robust controller synthesis of switched systems using counterexample guided framework," in *Proc. 13th Int. Conf. Embedded Softw.*, New York, NY, USA, ACM, 2016, pp. 8: 1–10:10.
[66] T. Koo, G. J. Pappas, and S. Sastry, "Mode switching synthesis for reachability specifications," in *Proc. 4th Int. Workshop Hybrid Syst., Comput. Control*, Rome, Italy, Mar. 2001, pp. 333–346.
[67] A. Taly, S. Gulwani, and A. Tiwari, "Synthesizing switching logic using constraint solving," *Int. J. Softw. Tools Technol. Transfer*, vol. 13, no. 6, pp. 519–535, Nov. 2011.
[68] S. Jha, S. A. Seshia, and A. Tiwari, "Synthesis of optimal switching logic for hybrid systems," in *Proc. 11th Int. Conf. Embedded Softw., Taipei, Taiwan, Oct. 9–14, 2011*, pp. 107–116.
[69] H. Zhao, N. Zhan, and D. Kapur, "Synthesizing switching controllers for hybrid systems by generating invariants," in *Proc. Theories Program. Formal Methods*, Apr. 2013, pp. 354–373.
[70] A. Fehnker and F. Ivančić, "Benchmarks for hybrid systems verification," in *Hybrid Systems: Computation and Control*, R. Alur and G. J. Pappas, Eds. Berlin, Heidelberg: Springer, 2004, pp. 326–341.
[71] T. Place and M. Zeitoun, "The tale of the quantifier alternation hierarchy of first-order logic over words," *ACM SIGLOG News*, vol. 2, no. 3, pp. 4–17, 2015.
[72] D. Monniaux, "A quantifier elimination algorithm for linear real arithmetic," in *Proc. Int. Conf. Log. Program. Artif. Intell. Reasoning*, Springer, 2008, pp. 243–257.
[73] G. E. Collins and H. Hong, "Partial cylindrical algebraic decomposition for quantifier elimination," *J. Symbolic Comput.*, vol. 12, no. 3, pp. 299–328, Sep. 1991.
[74] S. Bittanti, A. J. Laub, and J. C. Willems, *The Riccati Equation*. Berlin, Germany: Springer Science and Business Media, 2012.
[75] A. Richards and J. How, "Mixed-integer programming for control," in *Proc. Amer. Control Conf.*, 2005, pp. 2676–2683.
[76] C. H. Papadimitriou and M. Yannakakis, "The complexity of facets (and some facets of complexity)," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, May 1982, pp. 255–260.
[77] B. Dutertre, "Yices 2.2," in *Computer-Aided Verification (CAV'2014) (ser. Lecture Notes in Computer Science)*, A. Biere and R. Bloem, Eds., vol. 8559. Berlin, Germany: Springer, Jul. 2014, pp. 737–744.

Chuchu Fan received the bachelor's degree from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2019.

She is the Wilson Assistant Professor of Aeronautics and Astronautics with Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Prior to joining MIT in 2020, she was a Postdoctoral Researcher with the Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA. Her research interests include the areas of formal methods and control for safe autonomy.

Dr. Fan is a recipient of multiple prestigious awards including Young Researcher for Heidelberg Laureate Forum in 2017, Rising Stars in EECS in 2016, EMSOFT'16 Best Paper finalist, and Robert Bosch Best Verification Award in CPSWeek'15.

Zengyi Qin received the B.E. degree (with hons.) in electronic engineering from Tsinghua University, Beijing, China, in 2021. He is currently working toward the Graduate degree with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA , USA, under the supervision of Prof. Chuchu Fan.

During his undergraduate study, he also spent one year with Microsoft Research Asia. His research interests include safety autonomous systems and machine learning.

Umang Mathur received the Graduate degree from Indian Institute of Technology Bombay, Mumbai, India, in 2014. He currently working toward the Ph.D. degree with the Department of Computer Science, University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA.

He is currently advised by Prof. M. Viswanathan. He spent a year with WorldQuant Research (India) as a Quantitative Researcher prior to joining UIUC. His research interests include formal methods and logic and their applications to programming languages, software engineering, and cyber-physical systems.

Mr. Mathur was the recipient of the 2019 Google Ph.D. Fellowship for Programming Technology and Software Engineering.

Qiang Ning received the bachelor's degrees in electronic engineering and in economics from Tsinghua University, Beijing, China, in 2013, and the master's degree in biomedical imaging and the Ph.D. degree from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2016 and 2019, respectively.

He was Applied Scientist with Amazon. Before that, he was a Research Scientist with the AllenNLP team at the Allen Institute for AI (AI2) from 2019 to 2020. His research interests include natural language understanding and learning from indirect supervision.

Sayan Mitra received the bachelor's degree in electrical engineering from Jadavpur University, Kolkata, India, in 1999, the master's degree in computer science from the Indian Institute of Science, Bengaluru, India, in 2001, and the Doctorate degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2007.

He is currently a Professor of electrical and computer engineering, University of Illinois at Urbana-Champaign, Champaign, IL, USA. His textbook on verification of cyber-physical systems was published by MIT Press in 2021. His research interests include formal methods, hybrid systems, and safe autonomy.

Prof. Mitra is a recipient of the National Science Foundation's CAREER Award, Air Force Office of Scientific Research Young Investigator Program Award, and several best paper awards.

Mahesh Viswanathan received the bachelor's degree in computer science from the Indian Institute of Technology Kanpur, Kanpur, India, in 1995, and the doctorate degree from the University of Pennsylvania, Philadelphia, PA, USA, in 2000.

He was a Postdoctoral Fellow with DIMACS with a joint appointment with Telcordia Technologies in 2000–2001. Since Fall 2001, he has been on the faculty with the University of Illinois at Urbana-Champaign. His research interests include the core areas of logic, automata theory, and algorithm design, with applications to the algorithmic verification of systems.