Controller Synthesis Made Real: Reach-avoid Specifications and Linear Dynamics *

Chuchu Fan[1 [0000−0003−4671−233X]], Umang Mathur[1 [0000−0002−7610−0660]], Sayan Mitra[1 [0000−0001−7082−5516]], and Mahesh Viswanathan

{cfan10,umathur3,mitras,vmahesh}@illinois.edu
University of Illinois at Urbana Champaign, USA

Abstract. We address the problem of synthesizing provably correct controllers for linear systems with reach-avoid specifications. Our solution uses a combination of an open-loop controller and a tracking controller, thereby reducing the problem to smaller tractable problems. We show that, once a tracking controller is fixed, the reachable states from an initial neighborhood, subject to any disturbance, can be over-approximated by a sequence of ellipsoids, with sizes 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. Exploiting several techniques for tightening the over-approximations, we reduce the open-loop controller synthesis problem to satisfiability over quantifier-free linear real arithmetic. 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 present RealSyn, a tool implementing this synthesis algorithm, and we show that it scales to several high-dimensional systems with complex reach-avoid specifications.

1 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 not a new or a solved problem, but there has been resurgence of interest with the rise of powerful tools and compelling applications such as vehicle path planning [11], motion control [10,23], circuits design [30] and various other engineering areas.

In this paper, we study synthesis for linear, discrete-time, plant models with bounded disturbance — a standard view of control systems [17,3]. We will consider *reach-avoid* specifications which require that starting from any initial state

* This work is supported by the grant CCF 1422798 from the National Science Foundation.

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, multi-robot coordination, and spacecraft autonomy, and have been studied for linear, nonlinear, as well as stochastic models [7,18,14,9].

Textbook control design methods address specifications like stability, disturbance rejection, asymptotic convergence, but they do not provide formal guarantees about reach-avoid specifications. Another approach is based on *discrete abstraction*, where a discrete, finite-state, symbolic 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. Theoretically, these methods can be applied to systems with nonlinear dynamics and they can synthesize controllers for a general class of LTL specifications. However, in practice, the discretization step leads to a severe state space explosion for higher dimensional models. Indeed, we did not find any reported evaluation of these tools (see related work) on benchmarks that go beyond 5-dimensional plant models.

In this paper, the controller we synthesize, follows a natural paradigm for designing controllers. The approach is to first design 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 combined, that drives these other trajectories towards the trajectory starting from x_0.

However, designing such a combined controller can be computationally expensive [32] because of the interdependency between the open-loop controller and the tracking controller. Our secret sauce in making this approach feasible, is to demonstrate that the two controllers can be synthesized in a decoupled way. Our strategy is as follows. We first design a tracking controller using a standard control-theoretical method called LQR (linear quadratic regulator) [5]. The crucial observation 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 [24] centered around the reference trajectory. The size of these ellipsoidal sets is solely dependent on the tracking controller, and is independent of the reference trajectory or the open-loop controller. On the flip side, 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 in logic. The straightforward encoding of the synthesis problem 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 special properties of polytopes and hyper-rectangles, and reduce the original $\exists\forall$-formula into the quantifier-free fragment of linear arithmetic (QF-LRA).

Our overall algorithm (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 (Theorem 1), and for a class of robust linear systems, it is also complete (Theorem 2).

We have implemented the synthesis algorithm in a tool called RealSyn. Any SMT solver can be plugged-in for solving the open-loop problem; we present experimental results with Z3, CVC4 and Yices. We report the performance on 24 benchmark problems (using all three solvers). Results show that our approach scales well for complex models — including a system with 84-dimensional dynamics, another system with 3 vehicles (12-dimensional) trying to reach a common goal while avoiding collision with the obstacles and each other, and yet another system with 10 vehicles (20 dimensional) trying to maintain a platoon. RealSyn usually finds a controller within 10 minutes with the fastest SMT solver. The closest competing tool, Tulip [39,13], does not return any result even for some of the simpler instances.

Related Work We briefly review related work on formal controller synthesis according to the plant model type, specifications, and approaches.

Plants and specifications. In increasing order of generality, the types of plant models that have been considered for controller synthesis are double-integrator models [10], linear dynamical models [28,20,38,34], piecewise affine models [18,40], and nonlinear (possibly switched) models [31,7,33,25]. There is also a line of work on synthesis approaches for stochastic plants (see [1], and the references therein). With the exceptions noted below, most of these papers consider continuous time plant models, unlike our work.

There are three classes of specifications typically used for synthesis. In the order of generality, they are: (1) pure safety or invariance specifications [33,15,2], (2) reach-avoid [14,33,15,7,18], and (3) more general LTL and GR(1) [26,20,39] [38,40,16]. For each of these classes both bounded and unbounded-time variants have been considered.

Synthesis tools. There is a growing set of controller synthesis algorithms that are available as implemented tools and libraries. This includes tools like CoSyMa [27], Pessoa [30], LTLMop [37,22], Tulip [39,13], SCOTS [31], that rely on the computation of some sort of a discrete (or symbolic) abstraction. Our trial with a 4-dimensional example on Tulip [39,13] did not finish the discretization step in one hour. LTLMop [37,22] handles GR(1) LTL specifications, which are more general than reach-avoid specifications considered in this paper, but it is designed for 2-dimensional robot models working in the Euclidean plane. An alternative synthesis approach generates mode switching sequences for switched system models [29,21,35,19,41] 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 paper. Abate et. al. [2] use a controller template similar to the one considered in this paper for invariant specifications. A counter-example guided inductive synthesis (CEGIS) 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 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. Several of the benchmarks are adopted from [2].

2 Preliminaries and Problem Statement

Notation For a set A and a finite sequence σ in A^*, we denote the t^{th} element of σ by $\sigma[t]$. $\mathbb{R}^n$ is the n-dimensional Euclidean space. Given a vector $x \in \mathbb{R}^n$, $x(i)$ is the i^{th} component of x. We will use boldfaced letters (for example, $\mathbf{x}, \mathbf{d}, \mathbf{u}$, etc.,) to denote a sequence of vectors.

For a vector x, $x^\intercal$ is its transpose. Given an invertible matrix $M \in \mathbb{R}^{n\times n}$, $\|x\|_M \triangleq \sqrt{x^\intercal M^\intercal M x}$ is called the *M-norm* of x. For $M = I$, $\|x\|_M$ is the familiar 2-norm. Alternatively, $\|x\|_M = \|Mx\|_2$. For a matrix A, $A \succ 0$ means A is positive definite. Given two symmetric matrices A and B, $A \preceq B$ means $A - B$ is negative semi-definite. Given a matrix A and an invertible matrix M of the same dimension, there exists an $\alpha \geq 0$ such that $A^\intercal M^\intercal M A \preceq \alpha M^\intercal M$. Intuitively, α is the largest scaling factor that can be achieved by the linear transformation from x to Ax when using M for computing the norm, and can be found as the largest eigenvalue of the symmetric matrix $(MAM^{-1})^\intercal (MAM^{-1})$.

Given a vector $c \in \mathbb{R}^n$, an invertible matrix M, and a scalar value $r \geq 0$, we define $\mathcal{E}_r(c, M) \triangleq \{x \mid \|x - c\|_M \leq r\}$ to be the ellipsoid centered at c with radius r and shape M. $\mathcal{B}_r(c) \triangleq \mathcal{E}_r(c, I)$ is the ball of radius r centered at c. Given two vectors $c, v \in \mathbb{R}^n$, $\mathcal{R}_v(c) \triangleq \{x \mid \wedge_{i=1}^n\, c(i) - v(i) \leq x(i) \leq c(i) + v(i)\}$ is the rectangle centered at c with the length vector v. For a set $S \subseteq \mathbb{R}^n$, a vector $v \in \mathbb{R}^n$, and a matrix $M \in \mathbb{R}^{n\times n}$ we define $v \oplus S \triangleq \{x + v \mid x \in S\}$ and $M \otimes S \triangleq \{Mx \mid x \in S\}$. We say a set $S \subseteq \mathbb{R}^n$ is a polytope if there is a matrix $A^{m\times n}$ and a vector $b \in \mathbb{R}^m$ such that $S = \{x \mid Ax \leq b\}$, and denote by $vert(S)$ the set of vertices of S.

2.1 Discrete time linear control systems

An (n, m)-*dimensional discrete-time linear system* $\mathcal{A}$ is a 5-tuple $\langle A, B, \Theta, U, D\rangle$, where (i) $A \in \mathbb{R}^{n\times n}$ is called the *dynamic matrix*, (ii) $B \in \mathbb{R}^{n\times m}$ is called the *input matrix*, (iii) $\Theta \subseteq \mathbb{R}^n$ is a *set of initial states* (iv) $U \subseteq \mathbb{R}^m$ is the *space of inputs*, (v) $D \subseteq \mathbb{R}^n$ 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 $t > 0$,

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

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 $x \in \mathbb{R}^n$ at time t, the control input to the plant at time t is:

$$\mathbf{u}[t] = g(x_0, x). \tag{2}$$

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 (1) and (2) define a unique execution $\mathbf{x}$ of $\mathcal{A}$. A state x is *reachable in t-steps* if there exists an execution $\mathbf{x}$ of $\mathcal{A}$ such that $\mathbf{x}[t] = x$. The set of all reachable states from $S \subseteq \Theta$ 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 write $\mathsf{Reach}(S, T)$.

2.2 Bounded controller synthesis problem

Given a (n, m)-*dimensional 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 disturbance $\mathbf{d} \in D^T$, the unique execution $\mathbf{x}$ of $\mathcal{A}$ with g, starting from $\mathbf{x}[0] = \theta$ satisfies (i) for all $t \leq T$, $\mathbf{u}[t] \in U$, (ii) for all $t \leq T$, $\mathbf{x}[t] \notin \mathbf{O}[t]$, and (iii) $\mathbf{x}[T] \in G$.

For the rest of the paper, we will assume that each of the sets in $\{\mathbf{O}[t]\}_{t\in\mathbb{N}}$, G and U are closed polytopes. Moreover, we assume that the pair (A, B) is controllable [3].

Example *Consider a mobile robot that needs to reach the green area of an apartment starting from the entrance area, while avoiding the gray areas (Figure 1). The robot's dynamics is described by a linear model (for example the navigation model from [12]). The obstacle sequence* $\mathbf{O}$ *here is static, that is,* $\mathbf{O}[t] = \mathbf{O}[0]$ *for all* $t \geq 0$. *Both* Θ *and* G *are rectangles. Although these sets are depicted in 2D, the dynamics of the robot may involve a higher dimensional state space.*

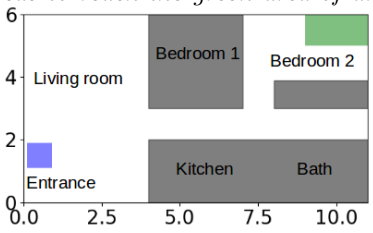

Fig. 1: The settings for controller synthesis of a mobile robot with reach-avoid specification.

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

3 Synthesis Algorithm

3.1 Overview

The controller synthesis problem requires one to find a state feedback controller that ensures that the trajectory 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 an effective representation. Thus, an "enumerative" representation, where a (separate) *open-loop controller* is constructed for each initial state, is not feasible — by an open-loop controller for initial state $x_0 \in \Theta$, we mean a control sequence $\mathbf{u}$ such that the corresponding execution $\mathbf{x}$ with $\mathbf{x}[0] = x_0$ and 0 disturbance satisfies the reach-avoid constraints. We, therefore, need a useful template that will serve as the representation for the feedback controller.

In control theory, one 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 [32] for reach-avoid specifications. The resulting trajectory under a combination of tracking controller plus reference trajectory can be described by the following system of equations.

$$\begin{aligned} &\mathbf{u}[t] = \mathbf{u}_{\mathsf{ref}}[t] + K(\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t]), \text{with} \\ &\mathbf{x}_{\mathsf{ref}}[t+1] = A\mathbf{x}_{\mathsf{ref}}[t] + B\mathbf{u}_{\mathsf{ref}}[t] \end{aligned} \tag{3}$$

The tracking controller is given by the matrix K that determines the additive component of the input based on the difference between the current state and the reference trajectory. Once $\mathbf{x}_{\mathsf{ref}}[0]$ and the open-loop control sequence $\mathbf{u}_{\mathsf{ref}}$ is fixed, the value of $\mathbf{x}_{\mathsf{ref}}[t]$ is determined at each time step $t \in \mathbb{N}$. Therefore, the controller g is uniquely defined by the tuple $\langle K, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$. We could rewrite the linear system in (3) as an augmented system

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

This can be rewritten as $\hat{\mathbf{x}}[t+1] = \hat{A}\hat{\mathbf{x}}[t] + \hat{B}\hat{\mathbf{u}}[t] + \hat{\mathbf{d}}[t]$. The closed-form solution is $\hat{\mathbf{x}}[t] = \hat{A}^t\hat{\mathbf{x}}[0] + \sum_{i=0}^{t-1} \hat{A}^{t-1-i}(\hat{B}\hat{\mathbf{u}}[i] + \hat{\mathbf{d}}[i])$. To synthesize a controller g of this form, therefore, requires finding $K, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}}$ such that the closed-form solution meets the reach-avoid specification. This is indeed the approach followed in [32], albeit in the continuous time setting. Observe that in the closed-form solution, $\hat{A}$, $\hat{\mathbf{u}}$, 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 paper, to achieve scalability, we take a slightly different approach than the one where $K, \mathbf{x}_{\mathsf{ref}}[0]$, and $\mathbf{u}_{\mathsf{ref}}$ are simultaneously synthesized. We first synthesize a tracking controller K, *independent* of $\mathbf{x}_{\mathsf{ref}}[0]$ and $\mathbf{u}_{\mathsf{ref}}$, using the standard LQR method. Once K is synthesized, we show that, no matter what $\mathbf{x}_{\mathsf{ref}}[0]$, and $\mathbf{u}_{\mathsf{ref}}$ are, the state of the system at time t starting from x_0 is guaranteed to be contained within an ellipsoid centered at $\mathbf{x}_{\mathsf{ref}}[t]$ and of radius that depends only on K, the initial distance between x_0 and $\mathbf{x}_{\mathsf{ref}}[0]$, time t, and disturbance. Moreover, this radius is only a *linear* function of the initial distance (Lemma 1). Thus, if we can synthesize an open-loop controller $\mathbf{u}_{\mathsf{ref}}$ starting from some state

$\mathbf{x}_{\mathsf{ref}}[0]$, such that ellipsoids centered around $\mathbf{x}_{\mathsf{ref}}$ 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 $\mathbf{x}_{\mathsf{ref}}[0]$. The radius of the ball around $\mathbf{x}_{\mathsf{ref}}[0]$ for which the controller is guaranteed to work, will depend on the radii of the ellipsoids around $\mathbf{x}_{\mathsf{ref}}$ that satisfy the reach-avoid specification. This decoupled approach to synthesis is the first key idea in our algorithm.

Following the above discussion, crucial to the success of the decoupled approach is to obtain a tight characterization of the radius of the ellipsoid around $\mathbf{x}_{\mathsf{ref}}[t]$ that contains the reach set, as a function of the initial distance — too conservative a bound will imply that the combined controller only works for a tiny set of initial states. The ellipsoid's shape and direction, which is characterized by a coordinate transformation matrix M, also affect the tightness of the over-approximations. We determine the shape and direction of the ellipsoids that give us the tightest over-approximation using an SDP solver (Section 3.4).

Synthesizing the tracking controller K, still leaves open the problem of synthesizing an open-loop controller for an initial state $\mathbf{x}_{\mathsf{ref}}[0]$. A straightforward encoding of the problem of synthesizing a open-loop controller, that works for all initial states in some ball around $\mathbf{x}_{\mathsf{ref}}[0]$, 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. The next key idea in our algorithm is to simplify these constraints. By exploiting special properties of polytopes and hyper-rectangles, we reduce the original $\exists\forall$-formula into the *quantifier-free* fragment of *linear* real arithmetic (QF-LRA) (Section 3.5).

Putting it all together, the overall algorithm (Algorithm 1) works as follows. After computing an initial tracking controller K, coordinate transformation M for optimal ellipsoidal approximation of reach-sets, 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 (Theorem 1)—whenever a controller is synthesized, it meets the specifications. Further, for robust systems (defined later in the paper), our algorithm is guaranteed to terminate when the system has a combined controller for all initial states (Theorem 2).

3.2 Synthesizing the tracking controller K

Given any open-loop controller $\mathbf{u}_{\mathsf{ref}}$ and the corresponding reference execution $\mathbf{x}_{\mathsf{ref}}$, by replacing in Equation (1) the controller of Equation (3) we get:

$$\mathbf{x}[t+1] = (A+BK)\mathbf{x}[t] - BK\mathbf{x}_{\mathsf{ref}}[t] + B\mathbf{u}_{\mathsf{ref}}[t] + \mathbf{d}[t]. \tag{4}$$

Subtracting $\mathbf{x}_{\mathsf{ref}}[t+1]$ 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 $\mathbf{x}_{\mathsf{ref}}$ changes with time as:

$$\mathbf{x}[t+1] - \mathbf{x}_{\mathsf{ref}}[t+1] = (A+BK)(\mathbf{x}[t] - \mathbf{x}_{\mathsf{ref}}[t]) + \mathbf{d}[t]. \tag{5}$$

With $A_c \triangleq A + BK$, $\mathbf{y}[t] \triangleq \mathbf{x}[t+1] - \mathbf{x}_{\mathsf{ref}}[t+1]$, Equation (5) becomes $\mathbf{y}[t+1] = A_c\mathbf{y}[t] + d[t]$. We want $\mathbf{x}[t]$ to be as close to $\mathbf{x}_{\mathsf{ref}}[t]$ as possible, which means K should be designed to make $|\mathbf{y}[t]|$ converge to 0. Equivalently, K should be designed as a linear feedback controller such that A_c is stable[1]. Such a matrix K can be computed using classical control theoretic methods. In this work, we compute K as a linear (stable) feedback controller using LQR as stated in the following proposition.

Proposition 1 (LQR). *For linear system $\mathcal{A}$ with (A, B) to be controllable and 0 disturbance, fix any $Q, R \succ 0$ and let $J \triangleq \mathbf{x}^\intercal[T]Q\mathbf{x}[T] + \sum_{i=0}^{T-1}(\mathbf{x}^\intercal[i]Q\mathbf{x}[i] + \mathbf{u}^\intercal[i]R\mathbf{u}[i])$ be the corresponding quadratic cost. Let X be the unique positive definite solution to the discrete-time Algebraic Riccati Equation (ARE): $A^\intercal XA - X - A^\intercal XB(B^\intercal XB + R)^{-1}B^\intercal XA + Q = 0$, and $K \triangleq -(B^\intercal XB + R)^{-1}B^\intercal XA$. Then $A + BK$ is stable, and the corresponding feedback input minimizes J.*

Methods for choosing Q and R are outside the scope of this paper. We fix Q and R to be identity matrices for most examples. Roughly, for a given R, scaling up Q results in a K that makes an execution $\mathbf{x}$ converge faster to the reference execution $\mathbf{x}_{\mathsf{ref}}$.

3.3 Reachset over-approximation with tracking controller

We present a method for over-approximating the reachable states of the system for a given tracking controller K (computed as in Proposition 1) and an open-loop controller $\mathbf{u}_{\mathsf{ref}}$ (to be computed in Section 3.5).

Lemma 1. *Consider any $K \in \mathbb{R}^{m\times n}$, any initial set $S \subseteq \mathcal{E}_{r_0}(\mathbf{x}_{ref}[0], M)$ and disturbance $D \subseteq \mathcal{E}_\delta(0, M)$, where $r_0, \delta \geq 0$ and $M \in \mathbb{R}^{n\times n}$ is invertible. For any open-loop controller $\mathbf{u}_{ref}$ and the corresponding reference execution $\mathbf{x}_{ref}$,*

$$\mathit{Reach}(S, t) \subseteq \mathcal{E}_{r_t}(\mathbf{x}_{ref}[t], M), \forall\ t \leq T, \tag{6}$$

where $r_t = \alpha^{\frac{t}{2}} r_0 + \sum_{i=0}^{t-1} \alpha^{\frac{i}{2}}\delta$, and $\alpha \geq 0$ is such that $(A+BK)^\intercal M^\intercal M(A+BK) \preceq \alpha M^\intercal M$.

Lemma 1 can be proved using the triangular inequality for the norm of Equation (5). From Lemma 1, it follows that given a open-loop controller $\mathbf{u}_{\mathsf{ref}}$ and the corresponding reference trajectory $\mathbf{x}_{\mathsf{ref}}$, the reachable states from $S \subseteq \mathcal{E}_{r_0}(\mathbf{x}_{\mathsf{ref}}[0], M)$ at time t can be over-approximated by an ellipsoid centered at $\mathbf{x}_{\mathsf{ref}}[t]$ with size $r_t \triangleq \alpha^{\frac{t}{2}} r_0 + \sum_{i=0}^{t-1} \alpha^{\frac{i}{2}}\delta$. Here M is any invertible matrix that defines the shape of the ellipsoid and it influences the value of α. As the over-approximation (r_t) grows exponentially with t, it makes sense to choose M in a way that makes α small. In next section, we discuss how M and α are chosen to achieve this.

3.4 Shaping ellipsoids for tight over-approximating hyper-rectangles

The choice of M and the resulting α may seem like a minor detail, but a bad choice here can doom the rest of the algorithm to be impractical. For example, if we fix M to be the identity matrix I, the resulting value of α may give

[1] $A + BK$ has spectral radius $\rho(A + BK) < 1$

over-approximations that are too conservative. Even if the actual executions are convergent to $\mathbf{x}_{\mathsf{ref}}$ the resulting over-approximation can exponentially blow up.

We find the smallest exponential convergence/divergence rate (α) by solving for P in the following semi-definite program (SDP):

$$\begin{aligned} \min_{P \succ 0, \alpha \in \mathbb{R}} \quad & \alpha \\ \text{s.t} \quad & (A+BK)^\intercal P (A+BK) \preceq \alpha P. \end{aligned} \tag{7}$$

This gives M as the unique matrix such that $P = M^T M$.

In the rest of the paper, the reachset over-approximations will be represented by hyper-rectangles to allow us to efficiently use the existing SMT solvers. That is, the ellipsoids given by Lemma 1 have to be bounded by hyper-rectangles. For any coordinate transformation matrix M, the ellipsoid with unit size $\mathcal{E}_1(0, M) \subseteq \mathcal{R}_v(0)$, with $v(i) = \min_{x \in \mathcal{E}_1(0,M)} x(i)$. This $v(i)$ is also computed by solving an SDP. Similarly, $\mathcal{E}_r(0, M) \subseteq \mathcal{R}_{rv}(0)$. Therefore, from Lemma 1, it follows that $\mathsf{Reach}(S, t) \subseteq \mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t])$ with $r_t = \alpha^{\frac{t}{2}} r_0 + \sum_{i=0}^{t-1} \alpha^{\frac{i}{2}} \delta$ and v is the size vector of the rectangle bounding $\mathcal{E}_1(0, M)$. These optimization problems for computing M, α, and v have to be solved once per synthesis problem.

Example *Continuing the previous example. Suppose robot is asked to reach the target set in* 20 *steps. Figure 2 shows the projection of the reachset on the robot's position with synthesized controller. The curves are the references executions* $\mathbf{x}_{ref}$ *from 2 initials cover and the rectangles are reachset over-approximations such that every execution of the system starting from each initial cover is guaranteed to be inside the rectangles at each time step.*

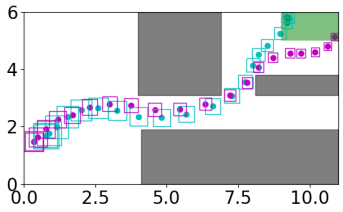

Fig. 2: Robot's position with the synthesized controllers using Algorithm 1.

3.5 Synthesis of open-loop controller

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

For synthesizing $\mathbf{u}_{\mathsf{ref}}$, we would like to formalize the problem in terms of constraints that will allow us to use SMT solvers. 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 efficiently 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.

Constraints for synthesizing $\mathbf{u}_{\mathsf{ref}}$. Let us fix an initial state x_0 and a radius r, defining a set of initial states $S = \mathcal{B}_r(x_0)$. The $\mathbf{u}_{\mathsf{ref}}$ synthesis problem can be stated as finding satisfying solutions for the formula $\phi_{\mathsf{synth}}(x_0, r)$.

$$\begin{aligned}\phi_{\mathsf{synth}}(x_0, r) \triangleq\ & \exists \mathbf{u}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}}[1], \dots \mathbf{u}_{\mathsf{ref}}[T-1], \\ & \exists \mathbf{x}_{\mathsf{ref}}[0], \mathbf{x}_{\mathsf{ref}}[1], \dots \mathbf{x}_{\mathsf{ref}}[T], \\ & \phi_{\mathsf{control}}(\mathbf{u}_{\mathsf{ref}}) \wedge \phi_{\mathsf{execution}}(\mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}, x_0) \\ & \wedge \phi_{\mathsf{avoid}}(x_0, r, \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) \wedge \phi_{\mathsf{reach}}(x_0, r, \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}})\end{aligned} \tag{8}$$

where ϕ_{control} constrains the space of inputs, $\phi_{\mathsf{execution}}$ states that the sequence $\mathbf{x}_{\mathsf{ref}}$ is a reference execution following Equation (3), ϕ_{avoid} specifies the safety constraint, ϕ_{reach} specifies that the system reaches G:

$$\begin{aligned}\phi_{\mathsf{control}}(\mathbf{u}_{\mathsf{ref}}) &\triangleq \bigwedge_{t=0}^{T-1} \mathbf{u}_{\mathsf{ref}}[t] \oplus \left(K \otimes \mathcal{R}_{r_t v}(0)\right) \subseteq U \\ \phi_{\mathsf{execution}}(\mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}, x_0) &\triangleq (\mathbf{x}_{\mathsf{ref}}[0] = x_0) \wedge \bigwedge_{t=0}^{T-1} (\mathbf{x}_{\mathsf{ref}}[t+1] = A\mathbf{x}_{\mathsf{ref}}[t] + B\mathbf{u}_{\mathsf{ref}}[t]) \\ \phi_{\mathsf{avoid}}(x_0, r, \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) &\triangleq \bigwedge_{t=0}^{T} \mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t]) \cap \mathbf{O}[t] = \varnothing \\ \phi_{\mathsf{reach}}(x_0, r, \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}}) &\triangleq \mathcal{R}_{r_T v}(\mathbf{x}_{\mathsf{ref}}[T]) \subseteq G.\end{aligned} \tag{9}$$

As discussed in Section 3.2, the vector v and the constants $r_0, \dots, r_T$ are precomputed using the radius r of the initial ball.

We make a few remarks about this formulation. First, each of the formulas $\phi_{\mathsf{control}}, \phi_{\mathsf{avoid}}$ and ϕ_{reach} represent sufficient conditions to check for the existence of $\mathbf{u}_{\mathsf{ref}}$. Second, the constraints stated above belong to the (decidable) theory of reals. However, $\phi_{\mathsf{control}}, \phi_{\mathsf{avoid}}$ 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 $\varphi_T(\cdot)$, $S \cap T = \varnothing$ corresponds to the formula $\forall x \cdot \neg(\varphi_S(x) \wedge \varphi_T(x))$ and $S \subseteq T$ (or equivalently $S \cap T^c = \varnothing$) corresponds to the formula $\forall x \cdot \varphi_S(x) \implies \varphi_T(x)$.

Reduction to QF-LRA. Since the sets G and U are bounded polytopes, G^c and U^c can be expressed as finite unions of (possibly unbounded) polytopes. Thus, the subset predicates $\mathbf{u}_{\mathsf{ref}}[t] \oplus \left(K \otimes \mathcal{R}_{r_t v}(0)\right) \subseteq U$ in ϕ_{control} and $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t]) \subseteq G$ in ϕ_{reach} can be expressed as a disjunction over finitely many predicates, each expressing the disjointness of two polytopes.

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 find a separating hyperplane that witnesses the disjointness of two polytopes. Let $P_1 = \{x \mid A_1 x \leq b_1\}$ and $P_2 = \{x \mid A_2 x \leq b_2\}$ be two polytopes such that P_1 is closed and bounded. Then, if there is an i for which each vertex v of P_1 satisfies $A_2^{(i)} v > b_2(i)$, we must have that $P_1 \cap P_2 = \varnothing$, where $A_2^{(i)}$ is the i^{th} row vector of the matrix A_2. That is, such a check is sufficient to ensure disjointness. Thus, in the formula ϕ_{avoid}, in order to check if $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t])$ does not intersect with $\mathbf{O}[t]$, we check if there is a face of the polytope $\mathbf{O}[t]$

such that all the vertices of $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t])$ lie on the other side of the face. The same holds for each of the inferred predicates in ϕ_{reach} and ϕ_{control}. Eliminating quantifiers is essential to scale our analysis to large high dimensional systems.

Further, when the set G has a hyper-rectangle representation, the containment check $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[T]) \subseteq G$ can directly be encoded as the conjunction of $O(n)$ linear inequalities, stating that for each dimension i, the lower and the upper bounds of $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t])$ in the i^{th} dimension, satisfy $l'_i \leq l_i \leq u_i \leq u'_i$, where l'_i and r'_i represent the bounds for G in the i^{th} dimension. Similarly, when $\mathbf{O}[t]$ has a rectangle representation, we can formulate the emptiness constraint $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t]) \cap \mathbf{O}[t] = \varnothing$ as $\bigvee_{i=1}^{n} (u_i < l'_i \vee l_i > u'_i)$, where l_i and u_i (resp. l'_i and u'_i) are the lower and upper bounds of $\mathcal{R}_{r_t v}(\mathbf{x}_{\mathsf{ref}}[t])$ (resp. $\mathbf{O}[t]$) in the i^{th} dimension. Since such simplifications can exponentially reduce the number of constraints generated, they play a crucial for the scalability.

The constraints for checking emptiness and disjointness, as discussed above, only give rise to linear constraints, do not have the $\forall$ quantification over states, and is a sound transformation of ϕ_{synth} into $\mathsf{QF\text{-}LRA}$. In Section 3.6 we will see that the reach set over-approximation can be made arbitrarily small when the disturbance is 0 by arbitrarily shrinking the size of the initial cover. Thus, these checks will also turn out to be sufficient to ensure that if there exists a controller, ϕ_{synth} is satisfiable.

Lemma 2. *Let $v \in \mathbb{R}^n$ and $r_0, \ldots, r_T \in \mathbb{R}$ be such that for any execution $\mathbf{x}_{ref}$ starting at x_0, we have $\forall t \leq T \cdot \mathsf{Reach}(\mathcal{B}_r(x_0), t) \subseteq \mathcal{R}_{r_t v}(\mathbf{x}_{ref}[t])$. If the formula $\phi_{synth}(x_0, r)$ is satisfiable, then there is a control sequence $\mathbf{u}_{ref}$ such that for every $x \in \mathcal{B}_r(x_0)$ and for every $\mathbf{d} \in \mathcal{D}^T$, the unique execution $\mathbf{x}$ defined by the controller $\langle K, x_0, \mathbf{u}_{ref} \rangle$ 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[2]. Indeed, in our experimental evaluation, we observed the downside of resorting to Farkas' Lemma in this problem.

3.6 Synthesis algorithm putting it all together

The presentation in Section 3.5 describes how to formalize constraints to generate a control sequence that works for a subset of the initial set Θ. The overall synthesis procedure (Algorithm 1), first computes a tracking controller K, then generates open-loop control sequences and reference executions in order to cover the entire set Θ.

The procedure bloatParams, computes a tracking controller K, a vector v and real valued parameters $\{c_1[t]\}_{t \leq T}$, $\{c_2[t]\}_{t \leq T}$, for the system $\mathcal{A}$ and time bound T with Q, R for the LQR method. Given any reference execution $\mathbf{x}_{\mathsf{ref}}$ and an initial set $\mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0])$, the parameters computed by bloatParams can be used

[2] 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

1: **Input**: $\mathcal{A}, T, \mathbf{O}[0], \ldots, \mathbf{O}[T], G, Q, R$
2: $r^* \leftarrow \mathsf{diameter}(\Theta)/2$
3: $K, v, c_1, c_2 \leftarrow$ BLOATPARAMS$(\mathcal{A}, T, Q, R)$
4: $\texttt{cover} \leftarrow \varnothing$
5: $\texttt{controllers} \leftarrow \varnothing$
6: **while** $\Theta \not\subseteq \texttt{cover}$ **do**
7: $\quad \psi_{\mathsf{synth}} \leftarrow$ GETCONSTRAINTS$(\mathcal{A}, T, \mathbf{O}[0], \ldots, \mathbf{O}[T], G, v, c_1, c_2, r^*, \texttt{cover})$
8: $\quad$ **if** CHECKSAT$(\psi_{\mathsf{synth}}) = \mathsf{SAT}$ **then**
9: $\quad\quad r, \mathbf{u}_{\mathsf{ref}}, \mathbf{x}_{\mathsf{ref}} \leftarrow$ MODEL(ψ_{synth})
10: $\quad\quad \texttt{cover} \leftarrow \texttt{cover} \cup \mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0])$
11: $\quad\quad \texttt{controllers} \leftarrow \texttt{controllers} \cup \{ (\langle K, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle , \mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0])) \}$
12: $\quad$ **else**
13: $\quad\quad r^* \leftarrow r^*/2$
14: **return** `controllers`;

to over-approximate $\mathsf{Reach}(\mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0]), t)$ with the rectangle $\mathcal{R}_{v'}(\mathbf{x}_{\mathsf{ref}}[t])$, where $v' = (c_1[t]r + c_2[t])v$. The computation of these parameters proceeds as follows. Matrix K is determined using LQR (Proposition 1). Now we use Equation (7) to compute the matrix M and the rate of convergence α. Vector v is then computed such that $\mathcal{E}_1(0, M)$ is bounded by $\mathcal{R}_v(0)$. Let $r_{\mathsf{unit}} = \max_{x \in \mathcal{B}_1(0)} \|x\|_M$ and $\delta = \max_{d \in \mathcal{D}} \|d\|_M$. Then we have, $\mathcal{B}_r(x_0) \subseteq \mathcal{E}_{r \cdot r_{\mathsf{unit}}}(x_0, M)$ for any x_0. The constants $c_1[0], \ldots c_1[T]$, $c_2[0], \ldots c_2[T]$ are computed as $c_1[t] = \alpha^{\frac{t}{2}} r_{\mathsf{unit}}$ and $c_2[t] = \sum_{i=0}^{t-1} \alpha^{\frac{i}{2}} \delta$; Section 3.2-Section 3.4 establish the correctness guarantees of these parameters. Clearly, these computations are independent of any reference executions $\mathbf{x}_{\mathsf{ref}}$ and control sequences $\mathbf{u}_{\mathsf{ref}}$.

The procedure GETCONSTRAINTS constructs the logical formula ψ_{synth} below such that whenever ψ_{synth} holds, we can find an initial radius r, and center x_0 in the set $\Theta \setminus \texttt{cover}$ and a control sequence $\mathbf{u}_{\mathsf{ref}}$ such that any controlled execution starting from $\mathcal{B}_r(x_0)$ satisfies the reach-avoid requirements.

$$\psi_{\mathsf{synth}} \triangleq \exists x_0 \, \exists r \cdot \Big(x_0 \in \Theta \wedge x_0 \notin \texttt{cover} \wedge r > r^* \wedge \phi_{\mathsf{synth}}(x_0, r) \Big) \tag{10}$$

Recall that the constants $r_0, \ldots r_T$ used in ϕ_{synth} are affine functions of r and thus ψ_{synth} falls in the QF-LRA fragment.

Line 8 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 9. The generated controller $\langle K, \mathbf{x}_{\mathsf{ref}}[0], \mathbf{u}_{\mathsf{ref}} \rangle$ is guaranteed to work for the ball $\mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0])$, which can be marked *covered* by adding it to the set `cover`. In order to keep all the constraints linear, one can further underapproximate $\mathcal{B}_r(\mathbf{x}_{\mathsf{ref}}[0])$ with the rectangle $\mathcal{R}_w(\mathbf{x}_{\mathsf{ref}}[0])$, where $w(i) = r/\sqrt{n}$ for each dimension $i \leq n$. If ψ_{synth} is unsatisfiable, then we reduce the minimum radius r^* (Line 13) and continue to look for controllers, until we find that $\Theta \subseteq \texttt{cover}$.

The set `controllers` is the set of pairs $(\langle K, x_0, \mathbf{u}_{\mathsf{ref}} \rangle, S)$, such that the controller $\langle K, x_0, \mathbf{u}_{\mathsf{ref}} \rangle$ 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 (Line 11). The following theorem asserts the soundness of Algorithm 1, and it follows from Lemma 1 and 2.

Theorem 1. *If Algorithm 1 terminates, then the synthesized controller is correct. That is, (a) for each $x \in \Theta$, there is a $(\langle K, x_0, \mathbf{u}_{ref}\rangle, S) \in$ `controllers`, such that $x \in S$, and (b) for each $(\langle K, x_0, \mathbf{u}_{ref}\rangle, S) \in$ `controllers`, the unique controller $\langle K, x_0, \mathbf{u}_{ref}\rangle$ is such that for every $x \in S$ and for every $\mathbf{d} \in D^T$, the unique execution defined by $\langle K, x_0, \mathbf{u}_{ref}\rangle$ and $\mathbf{d}$, 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 non-determinism 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 $\mathcal{A} = \langle A, B, \Theta, U, D\rangle$ is said to be ε-robustly controllable ($\varepsilon > 0$) with respect to the reach-avoid specification ($\mathbf{O}$, G) and matrix K, if (a) $D = \{0\}$, and (b) for every initial state $\theta \in \Theta$ and for every 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, then with the controller $\langle K, \theta, \mathbf{u}_{\mathsf{ref}}\rangle$ defined as in Equation (3), $\forall t \leq T, \mathsf{Reach}(\mathcal{B}_\varepsilon(\theta), t) \cap \mathbf{O}[t] = \varnothing$ and $\mathsf{Reach}(\mathcal{B}_\varepsilon(\theta), T) \subseteq G$, i.e., $\forall x \in \mathcal{B}_\varepsilon(\theta)$, the unique trajectory $\mathbf{x}$ defined by the controller $\langle K, \theta, \mathbf{u}_{\mathsf{ref}}\rangle$ starting from x also satisfies the reach avoid specification.

Theorem 2. *Let $\mathcal{A}$ be ε-robust with respect to the reach-avoid specification $(\mathbf{O}, G)$ and K, for some $\varepsilon > 0$. If there is a controller for $\mathcal{A}$ that satisfies the reach-avoid specification, then Algorithm 1 terminates.*

When the system is robust, then (in the absence of any disturbance i.e., $D = \{0\}$), the sizes $r_0, r_1, \ldots, r_T$ of the hyper-rectangles that overapproximate reach-sets go arbitrarily close to 0 as the initial cover converges to a single point (as seen in Lemma 1). Therefore, the over-approximations can be made arbitrarily precise as r^* decreases. Moreover, as r^* approaches 0, Equation (9) (with simplifications for QF-LRA), also becomes satisfiable whenever there is a controller. The correctness of Theorem 2 follows from both these observations.

4 RealSyn implementation and evaluation

4.1 Implementation

We have implemented our synthesis algorithm in a tool called RealSyn. RealSyn is written in Python. For solving Equation (10) it can interface with any SMT solver through Python APIs. We present experimental results with Z3 (version 4.5.1) [6], Yices (version 2.5.4) [8], and CVC4 (version 1.5) [4]. RealSyn leverages the incremental solving capabilities of these solvers as follows: The constraints ψ_{synth} generated (line 8 in Algorithm 1) can be expressed as $\exists x_0, \exists r \cdot \psi_1 \wedge \psi_2$, where $\psi_1 \triangleq \phi_{\mathsf{synth}}(x_0, r)$ and $\psi_2 \triangleq x_0 \in \Theta \wedge x_0 \notin$ `cover` $\wedge r > r^*$.

Since the bulk of the formula $\phi_{\mathsf{synth}}(x_0, r)$ is in ψ_1 and it does not change across iterations, we can generate this formula only once, and push it on the context stack of the solvers. The formula ψ_2 is different across iterations, and can be pushed and popped out of the stack as required. This minimizes the time taken for generation of constraints.

4.2 Evaluation

We use 24 benchmark examples[3] to evaluate the performance of RealSyn with three different solvers on a standard laptop with Intel® Core™ i7 processor, 16GB RAM, running Ubuntu 16.04. The results are reported in Table 1. The results are encouraging and demonstrate the effectiveness of using our approach and the feasibility of scalable controller synthesis for high dimensional systems and complex reach-avoid specifications.

Table 1: Controller synthesis using RealSyn and different SMT solvers. An explanation for the * marked entries can be found in Section 4.

	Model	n	m	Z3		CVC4		Yices	
				#iter	time(s)	#iter	time(s)	#iter	time(s)
1	1-robot	2	1	9	0.21	1	0.06	7	0.06
2	2-robot	4	2	164	12.62	11	0.31	183	2.26
3	running-example	4	2	N/A	T/O	N/A	T/O	1	319.97
4	1-car dynamic avoid	4	2	9	53.17	1	96.43	12	8.49
5	1-car navigation	4	2	18	7.49	1	3.05	17	6.73
6	2-car navigation	8	4	1	60.14	1	2668.2	1	4.07
7	3-car navigation	12	6	1	733.42	1	481.88	1	741.73
8	4-car platoon	8	4	1	0.37	1	0.21	1	0.15
9	8-car platoon	16	8	1	23.02	1	1.44	1	0.62
10	10-car platoon	20	10	1	459.36	1	20.93	1	7.74
11	example	3	1	82	2.32	18	0.10	67	0.43
12	cruise	1	1	1	0.06	1	0.03	1	0.02
13	motor	2	1	1	0.10	1	0.06	1	0.03
14	helicopter	3	1	81	2.31	13	0.08	70	0.38
15	magnetic suspension	2	1	39	0.47	2	0.05	39	0.08
16	pendulum	2	1	30	0.32	8	0.05	42	0.07
17	satellite	2	1	40	0.46	5	0.05	32	0.06
18	suspension	4	1	1	0.17	1	0.11	1	0.09
19	tape	3	1	1	0.12	1	0.07	1	0.07
20	inverted pendulum	2	1	39	0.49	2	0.05	39	0.09
21	magnetic pointer	3	1	44	1.12	12	0.08	134	0.83
22	helicopter	28	6	N/A (1*)	T/O (650*)	1	651.21	N/A	T/O
23	building	48	1	1 (1*)	1936.03 (240*)	N/A	T/O	1	552.48
24	pde	84	1	N/A (1*)	T/O (1800*)	1	8.48	1	8.87

Comparison with other tools. We considered other controller synthesis tools for possible comparison with RealSyn. In summary, CoSyMa [27], Pessoa [30], and SCOTS [31] do not explicitly support discrete-time sytems. LTLMop [37,22] is designed to analyze robotic systems in the (2-dimensional) Euclidean plane

[3] The examples are available at `http://umathur3.web.engr.illinois.edu/realsyn.html`

and thus not suitable for most of our examples. TuLiP [39,13] 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 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-dimensional system '1-robot' (Table 1), and returned `unrealizable`. On the benchmark '2-robot' ($n = 4$), TuLip did not return any answer within 1 hour. We checked these findings with the developers and they concurred that it is typical for TuLip to take hours even for 4-dimensional systems.

Benchmarks. Our benchmarks and their SMT encodings, could be of independent interest to the verification and SMT-community. Examples 1-10 are vehicle motion planning examples we have designed with reach-avoid specifications. Benchmarks 1-2 model robots moving on the Euclidean plane, where each robot is a 2-dimensional system and admits a 1-dimensional 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 [12]. Each vehicle is a 4-dimensional system with 2-dimensional input. Benchmark 3 is the system as our running example. Benchmark 4 describes one *ego* vehicle running on a two-lane road, trying to overtake a vehicle in front of it. The second vehicle serves as the 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.

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 [32]. Each vehicle is a 2-dimensional system with 1-dimensional input. For the 4-car platoon model, the running times reported in Table 1 are much smaller than the time (5 minutes) reported in [32]. This observation aligns with our analysis in Section 3.1.

Benchmarks 11-21 are from [2]. The specification here is that the reach set has to be within a safe rectangle (that is, $G = \mathit{true}$). In [2] each model is discretized using 8 different time steps and here we randomly pick one for each model. In general, the running time of RealSyn is less than those reported in [2] (their reported machine had better configuration). On the other hand, the synthesized controller from [2] considers quantization errors, while our approach does not provide any guarantee for that.

Benchmarks 22-24 are a set of high dimensional examples adopted and discretized from [36]. Similar to previous ones, the only specification is that the reach sets starting from an initial state with the controller should be contained within a safe rectangle.

Synthesis performance. In Table 1, 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. We specify a time limit of 1 hour and report T/O (timeout) for benchmarks that do not finish within this limit. All benchmarks are synthesized for a specification with $10 - 20$ steps.

In general, for low-dimensional systems (for example, in Benchmarks 11-21), each of the solvers finish quickly (in less than 1s), with CVC4 and Yices outperforming Z3 on most benchmarks. The Yices solver is faster than the other two on most examples. Z3 was the slowest on most, except a few (e.g., Benchmark 3,6) where CVC4 was much slower. The running time, in general, increases with the increase of the dimensionality but this relationship is far from simple. For example, the 84-dimensional Benchmark 24 was synthesized in less than 9 seconds by both CVC4 and Yices, possibly because the safety specification is rather simple for this problem.

The three solvers use different techniques for solving QF-LRA formulae with support for incremental solving. The default tactic in Z3 is such that it spends a large chunk of time when a constraint is pushed to the solver stack. In fact, for Benchmark 24, while the other two solvers finish within 9 seconds, Z3 did not finish pushing the constraints in the solver stack. When we disable incremental solving in Z3, the Benchmarks 22, 23 and 24 finish in about 650, 240 and 1800 seconds respectively (marked with *). The number of iterations widely vary across solvers, with CVC4 usually finishing in the fewest number of iterations. Despite the larger number of satisfiability queries, Yices manages to finish close to CVC4 on most examples.

5 Conclusion

We proposed a novel technique for synthesizing controllers for systems with 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 demonstrates the value of the approach when analyzing systems with complex dynamics and specifications.

There are several avenues for future work. This includes synthesis of combined controllers for nonlinear dynamical and hybrid systems, and for more general temporal logic specifications. Generating witnesses to show the absence of controllers is also an interesting direction.

References

1. Alessandro Abate, Saurabh Amin, Maria Prandini, John Lygeros, and Shankar Sastry. Computational approaches to reachability analysis of stochastic hybrid systems. In *Hybrid Systems: Computation and Control, 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007, Proceedings*, pages 4–17, 2007.
2. Alessandro Abate, Iury Bessa, Dario Cattaruzza, Lucas C. Cordeiro, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. Automated formal synthesis of digital controllers for state-space physical plants. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 462–482, 2017.
3. Panos J Antsaklis and Anthony N Michel. *A linear systems primer*, volume 1. Birkhäuser Boston, 2007.
4. Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. Cvc4. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, pages 171–177, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
5. Stephen Boyd and Lieven Vandenberghe. Convex optimization. 2004.
6. Leonardo De Moura and Nikolaj Bj**o**rner. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.
7. Jerry Ding and Claire J. Tomlin. Robust reach-avoid controller synthesis for switched nonlinear systems. In *Proceedings of the 49th IEEE Conference on Decision and Control, CDC 2010, December 15-17, 2010, Atlanta, Georgia, USA*, pages 6481–6486, 2010.
8. Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, July 2014.
9. Peyman Mohajerin Esfahani, Debasish Chatterjee, and John Lygeros. The stochastic reach-avoid problem and set characterization for diffusions. *Automatica*, 70:43–56, 2016.
10. Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.
11. Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Hybrid controllers for path planning: A temporal logic approach. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4885–4890. IEEE, 2005.
12. Ansgar Fehnker and Franjo Ivan**v**ćić. Benchmarks for hybrid systems verification. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control*, pages 326–341, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
13. Ioannis Filippidis, Sumanth Dathathri, Scott C. Livingston, Necmiye Ozay, and Richard M. Murray. Control design for hybrid systems with tulip: The temporal logic planning toolbox. In *2016 IEEE Conference on Control Applications, CCA 2016, Buenos Aires, Argentina, September 19-22, 2016*, pages 1030–1041, 2016.
14. Jaime F. Fisac, Mo Chen, Claire J. Tomlin, and S. Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, pages 11–20, 2015.

15. Antoine Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.
16. Ebru Aydin Gol, Mircea Lazar, and Calin Belta. Language-guided controller synthesis for linear systems. *IEEE Trans. Automat. Contr.*, 59(5):1163–1176, 2014.
17. Joao P Hespanha. *Linear systems theory*. Princeton university press, 2009.
18. Zhenqi Huang, Yu Wang, Sayan Mitra, Geir E. Dullerud, and Swarat Chaudhuri. Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*, pages 7434–7439, 2015.
19. Susmit Jha, Sanjit A. Seshia, and Ashish Tiwari. Synthesis of optimal switching logic for hybrid systems. In *Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011, part of the Seventh Embedded Systems Week, ESWeek 2011, Taipei, Taiwan, October 9-14, 2011*, pages 107–116, 2011.
20. Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Automat. Contr.*, 53(1):287–297, 2008.
21. Tak-John Koo, George J. Pappas, and Shankar Sastry. Mode switching synthesis for reachability specifications. In *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, pages 333–346, 2001.
22. Hadas Kress-Gazit, Gerogios E. Fainekos, and George J. Pappas. Temporal logic based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.
23. Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):null, 2018.
24. Alex A Kurzhanskiy and Pravin Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1):26–38, 2007.
25. Jun Liu, Necmiye Ozay, Ufuk Topcu, and Richard M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Trans. Automat. Contr.*, 58(7):1771–1785, 2013.
26. Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Compositional synthesis of finite state abstractions. *CoRR*, abs/1612.08515, 2016.
27. Sebti Mouelhi, Antoine Girard, and Gregor Gössler. Cosyma: A tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC '13, pages 83–88, New York, NY, USA, 2013. ACM.
28. Mustapha Ait Rami and Fernando Tadeo. Controller synthesis for positive linear systems with bounded controls. *IEEE Trans. on Circuits and Systems*, 54-II(2):151–155, 2007.
29. Hadi Ravanbakhsh and Sriram Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT '16, pages 8:1–8:10, New York, NY, USA, 2016. ACM.
30. Pritam Roy, Paulo Tabuada, and Rupak Majumdar. Pessoa 2.0: A controller synthesis tool for cyber-physical systems. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, pages 315–316, New York, NY, USA, 2011. ACM.
31. Matthias Rungger and Majid Zamani. Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th International Conference on Hybrid Systems:*

Computation and Control, HSCC '16, pages 99–104, New York, NY, USA, 2016. ACM.

32. Bastian Schürmann and Matthias Althoff. Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *2017 American Control Conference, ACC 2017, Seattle, WA, USA, May 24-26, 2017*, pages 2522–2529, 2017.
33. Paulo Tabuada. *Verification and Control of Hybrid Systems - A Symbolic Approach*. Springer, 2009.
34. Paulo Tabuada and George J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Trans. Automat. Contr.*, 51(12):1862–1877, 2006.
35. Ankur Taly, Sumit Gulwani, and Ashish Tiwari. Synthesizing switching logic using constraint solving. *STTT*, 13(6):519–535, 2011.
36. Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. Large-scale linear systems from order-reduction. In *ARCH@CPSWeek 2016, 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems, Vienna, Austria*, pages 60–67, 2016.
37. Kai Weng Wong, Cameron Finucane, and Hadas Kress-Gazit. Provably-correct robot control with ltlmop, OMPL and ROS. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, page 2073, 2013.
38. Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Trans. Automat. Contr.*, 57(11):2817–2830, 2012.
39. Tichakorn Wongpiromsarn, Ufuk Topcu, Necmiye Ozay, Huan Xu, and Richard M. Murray. Tulip: A software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, pages 313–314, New York, NY, USA, 2011. ACM.
40. Boyan Yordanov, Jana Tumova, Ivana Cerna, Jiri Barnat, and Calin Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Trans. Automat. Contr.*, 57(6):1491–1504, 2012.
41. Hengjun Zhao, Naijun Zhan, and Deepak Kapur. Synthesizing switching controllers for hybrid systems by generating invariants. In *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, pages 354–373, 2013.