Nonlinear Controller Synthesis and Automatic Workspace Partitioning for Reactive High-Level Behaviors

Jonathan A. DeCastro
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY, USA
jad455@cornell.edu

Hadas Kress-Gazit
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY, USA
hadaskg@cornell.edu

ABSTRACT

Motivated by the provably-correct execution of complex reactive tasks for robots with nonlinear, under-actuated dynamics, our focus is on the synthesis of a library of low-level controllers that implements the behaviors of a high-level controller. The synthesized controllers should allow the robot to react to its environment whenever dynamically feasible given the geometry of the workspace. For any behaviors that cannot guarantee the task given the dynamics, such behaviors should be transformed into dynamically-informative revisions to the high-level task. We therefore propose a framework for synthesizing such low-level controllers and, moreover, offer an approach for re-partitioning and abstracting the system based on the synthesized controller library.

We accomplish these goals by introducing a synthesis approach that we call *conforming funnels*, in which controllers are synthesized with respect to the given high-level behaviors, the geometrical constraints of the workspace, and a robot dynamics model. Our approach computes controllers using a verification approach that optimizes over a wide range of possible controllers to guarantee the geometrical constraints are satisfied. We also devise an algorithm that uses the controllers to re-partition the workspace and automatically adapt the high-level specification with a new discrete abstraction generated on these new partitions. We demonstrate the controllers generated by our synthesis framework in an experimental setting with a KUKA youBot executing a box transportation task.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.9 [**Artificial Intelligence**]: Robotics

Keywords

motion planning, controller synthesis, nonlinear systems, verification, barrier certificates

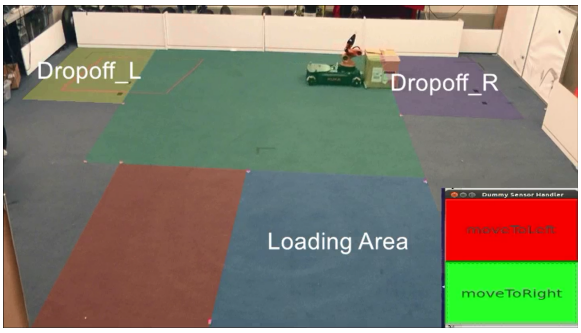


Figure 1: A robot executing a user-assisted warehouse supply task.

1. INTRODUCTION

Reactive synthesis frameworks have shown promise for robotics applications requiring guaranteed behaviors in unpredictable environments [27, 14]. From a high-level specification, a controller is automatically synthesized, without needing to code each behavior by hand. This enables sensor-rich systems to execute tasks in a provably-correct manner. For the workspace shown in Fig. 1, reactive synthesis addresses tasks such as: "If you see a box, move to the pick-up location. If you are carrying a box, then visit a requested drop-off location. Always avoid people."

The existence of such hybrid (mixed discrete- and continuous-state) controllers relies on the the ability to extend high-level guarantees (on a discrete abstraction of the system) to the low-level (continuous) dynamical system. While recent activity toward this end has focused on frameworks that use a combination of on-line and off-line approaches for correct-by-construction controller synthesis for nonlinear systems (e.g. [18, 26, 16, 27, 2, 9]), we aim to provide *a-priori* guarantees (i.e. at synthesis time) that the task is implementable on the physical system.

One of the goals of this work is an approach that synthesizes continuous controllers for a dynamical system to implement the behaviors of a finite state machine in response to a dynamic environment (factors outside the robot's control) given the geometrical constraints of the workspace. We build on an existing framework that implements high-level behaviors and verifies the resulting controllers over a wide

range of possible states, given a pre-defined low-level controller [9]. A high-level behavior consists of atomic actions that implement the transitions of a state machine, possibly in reaction to sensor events (e.g. motion between certain regions of the workspace, while avoiding other regions). To address the potential for conservatism of existing approaches that return a verified set of system states given a specific controller and system model (e.g. [9]), we aim for an approach that accepts a workspace, trajectory and nonlinear system model, and returns a feedback controller in addition to a verified set of continuous states.

If no such controllers can be found, there are a number of ways to revise the task in such a way that low-level controllers can be computed. For instance, if the synthesized controllers are unable to verify the act of moving to the pick-up location and avoid collisions with a person, then the statement "A person should never appear in the environment" could be added to the specification. However, guarantees without environment reactivity may be overly conservative on the part of the robot. Therefore, our second goal is to generate dynamically-informative revisions to the high-level task, alerting the user to any conditions – specifically, assumptions on what the environment is allowed to do – that are sufficient for the task to succeed. We aim for a framework that is able to automatically compute partitions in a manner that is consistent with the dynamics, resulting in more specific user feedback such as "When heading to a drop-off location, assume that a person will not appear when within 1 meter of the drop-off location".

The first main contribution of this paper is a new reachability-based controller design procedure, termed *conforming funnels* that takes into consideration a high-level behavior and the geometrical constraints of a workspace, providing a certificate (funnel) that the controller will implement the desired high-level behavior. We do so by introducing a new synthesis algorithm that combines the advantages of the invariant funnels approach [22] for nonlinear system verification and control barrier functions [20] to provide the ability to "shape" the feedback controller, and hence the resulting funnel, according to the workspace geometry. We provide a computational framework for automating the generation of such controllers as part of a library of controllers that are composed together to execute a high-level task. The resulting funnel conforms tightly to geometrical constraints where necessary, enabling the system to appropriately react to the environment while guaranteeing that the dynamical system adheres to all required high-level behaviors (e.g. collision avoidance). We do not address the challenge of finding optimal trajectories; hence we do not claim completeness in the sense that the framework will not necessarily find controllers wherever dynamically feasible.

Our second main contribution is a technique for generating a finer partitioning of the existing workspace based on an existing low-level controller that cannot implement a particular high-level behavior. New workspace regions that result from such a partitioning carry with them information concerning the verified behaviors of the dynamical system when the synthesized low-level controllers are executed at runtime. For instance, a partitioning may be created if, in the above example, a controller cannot verify collision-free motion for all trajectories that execute a robot's motion to the drop-off location. The new region for this case may indicate the set of states where the robot is unable to respond the appearance of a person when activating a certain low-level controller. We contribute a method for reasoning about such partitions via incremental changes to a discrete abstraction to make it consistent with the verified low-level controllers. In the context of a scheme developed recently for automatically synthesizing revisions to reactive specifications, [8], our proposed approach to finding revisions to the high-level specification may be viewed as a generalization when the underlying discrete abstraction depends on the result of low-level controller synthesis.

The dynamics-based partitioning approach can be displayed visually to users as an aid in explaining the implications of such revisions on the allowed environment behaviors. In this work, we do this by way of a labeled workspace map, which visually complements verbal statements explaining the revisions, which is demonstrated via both simulated and physical experiments on an actual robot. We also note that the creation of such abstractions (and hence revisions) is uniquely tied to our approach to low-level synthesis. With our proposed conforming funnels approach, we generate controllers that ensure that reactive behaviors are dynamically feasible. We evaluate for a specific example, the proposed synthesis framework against an existing approach to low-level controller synthesis. Furthermore, we show, through physical experiments on a KUKA youBot, we show that the trajectories generated by the physical platform remain within the funnels computed using our approach.

2. RELATED WORK

Numerous works have studied approaches to high-level planning that explicitly take into consideration the dynamics of an arbitrary nonlinear system. For static environments, Wolff, Topcu and Murray [26] synthesize a trajectory of control inputs by solving a constrained reachability problem that satisfies a given specification. Bhatia, et. al. [3] and Maly, et. al. [18] introduce two variants of a multi-layered synthesis paradigm applicable to nonlinear dynamics. In those approaches, obstacles that are unaccounted for at synthesis time are addressed by updating the planner when such obstacles are encountered at runtime. An abstraction-based approach is introduced in Wongpiromsarn, Topcu and Murray [27] that deals with nonlinear dynamics by formulating the synthesis task as a receding-horizon control problem. Our approach differs in that we rely on controllers constructed ahead of time with guarantees that the controllers can be executed over a wide range of continuous states. Rather than updating the controller on-the-fly to account for dynamics, in our approach we identify any necessary changes to the abstraction at synthesis time, providing an ability to alert the user to any necessary revisions to the specification.

A number of existing works address the reachability-based controller design problem for nonlinear systems. Works such as Tomlin, Lygeros and Sastry [23] and Mitchell, Bayen and Tomlin [19] are among those that solve the reach-avoid problem in a resolution-complete manner, while Burridge, Rizzi and Kodichek [5], Tedrake, et. al. [22] and Maidens and Arcak[17] offer methods that are generally not complete, but for which a solution can be found quickly for reasonable state dimensions. While those in the former category generally treat systems up to degree four, we opt for the trajectory-based approaches in the latter category that are capable of handling larger state dimensions. Specifically, our controller

synthesis procedure is based on the invariant funnels method introduced in [22]. The general approach has been shown to verify a variety of nonlinear systems, including those having up to 12 states. Conner, Rizzi and Choset [7] introduce an approach for sequencing together a set of verified motion primitives satisfying the transitions of a state machine encoding a non-reactive task. To accommodate *reactive* tasks, DeCastro and Kress-Gazit [9] introduce an algorithm for generating a library of atomic (low-level) controllers for a wide range of nonlinear systems satisfying the transitions of a finite state machine. The synthesis step can be repeated an arbitrary number of times in order to add to the verified space.

Our proposed conforming funnels approach is closely related to the control barrier functions (CBF) approach introduced by Wieland and Allgower [25]. The CBF approach uses synthesized functions known as barrier functions [20] to construct a controller for a nonlinear system that yields trajectories that avoid unsafe regions of the state space. The approach was later unified with control Lyapunov functions (CLF) by Romdlony and Jayawardhan [21] by extending the properties of such controllers to assure asymptotic convergence to a stabilizing point in addition to having guarantees for safety. We use these basic constructions in our work to locally modify a funnel by first computing regions of attraction to points along a trajectory without consideration to the problem constraints, and then compute a CBF to assemble a low-level controller that can guarantee invariance to any unsafe regions with respect to a specified initial set.

Our framework for computing partitions of given workspace map departs from existing works (e.g. Kloetzer and Belta [13]) in that we deal with tasks that are reactive in nature, whereas exiting works generally do not consider reactive environments at synthesis time. The workspace partitioning scheme developed in this work can be used in conjunction with existing abstraction update procedures in the literature (e.g. Clarke [6] and Liu and Ozay [15]), and hence can be seen as complementary to those works. Automatic generation of discrete abstractions have also been introduced recently in DeCastro, Raman and Kress-Gazit [10]. In the proposed approach, we specifically focus on the problem of adding dynamics-informed partitions, whereas in that work, the existing partitioning was used as the basis for constructing a new abstraction. When used for reactive synthesis, our dynamically-feasible synthesis framework may be viewed as an instance of *counterexample-guided inductive synthesis* (CEGIS) [1], where an abstraction is synthesized on the basis of the computed atomic controllers and their reachable sets (the counterexample).

3. PRELIMINARIES

We define several basic concepts that lay the foundation for the remainder of this paper: the finite state machine, trajectory stabilizing controller, and the invariant funnels method.

In this paper, we consider systems of the control-affine form

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where $x \in \mathbb{R}^n$ is the continuous state of the robot, $u \in \mathbb{R}^m$ the command input of the robot at time $t \in \mathbb{R}_{\geq 0}$ and f and g are smooth, continuous vector fields with respect to x. Note that a wide variety of robot dynamics (mobile robots, manipulators, walking robots) fit within this system class.

Notation: Given some $T \in \mathbb{R}_{\geq 0}$ and a function $\mathcal{L}(t) \subset \mathbb{R}^n$ defined for all $t \in [0, T]$, denote the set cover by $\mathcal{L}(t)$ for all $t \in [0, T]$ as $[\mathcal{L}]$. In particular, $[\mathcal{L}] = \{x \mid \exists t \in [0, T] \text{ s.t. } x \in \mathcal{L}(t)\}$.

3.1 Finite State Machine

For the purposes of our discussion, we assume in this work we are given a finite-state machine $\mathcal{A} = (AP_e, AP_s, S, S_0, \delta)$, where AP_e and AP_s are sets of atomic propositions corresponding, respectively, to the *environment* (sensed events that the robot must react to) and *system* (the actions of the robot); S is the set of (discrete) states; $S_0 \subseteq S$ is the set of initial states; and $\delta \subseteq S \times 2^{AP_e} \times S$ represents a state transition relation of the current state, the current value of the environment input, and the successor state, respectively.

For each state $s_i \in S$, let $X_i \subset \mathbb{R}^n$ be a continuous set in a space of dimension n associated with s_i. Also, denote ∂X_i to be the boundary of X_i.

3.2 Trajectory-Stabilizing Control

Given an initial state $x(0) \in X_0 \subset \mathbb{R}^n$, where X_0 is the initial set and a time horizon $T > 0$, denote $\xi : [0, T] \to \mathbb{R}^n$ as a continuous finite-time trajectory of states under $\dot{x} = f(x) + g(x)\mu(t)$. Here, $\mu : [0, T] \to \mathbb{R}^m$ is a trajectory of control inputs. To generate the trajectory $\xi(t)$ and determine the final time T, we use a feedback linearization-based control strategy to drive the system to a point in Cartesian space. Trajectories may also be found using trajectory optimization.

About $\xi(t)$, take the linearization of (1) to be $\dot{\tilde{x}} = A_\xi(t)\tilde{x}(t) + B_\xi(t)\tilde{u}(t)$, where $\tilde{x}(t) = x(t) - \xi(t)$ and $\tilde{u}(t) = u(t) - \mu(t)$. For a given trajectory and a linearization of the system, our task is to design a trajectory-stabilizing controller. We do so by solving the time-varying LQR problem [12] for the linearized system, whose result yields the full-state feedback control input $u(t) = K(t)(x(t) - \xi(t)) + \mu(t)$. Under this feedback, we denote the closed-loop system as

$$\dot{x} = f(x) + g(x)\left[K(t)(x(t) - \xi(t)) + \mu(t)\right] = \hat{f}(x, t), \tag{2}$$

where the closed-loop system $\hat{f}(x, t)$ is defined for all $t \in [0, T]$.

3.3 Invariant Funnels

We now summarize the invariant funnels method of [22], which takes as input a trajectory and stabilizing controller introduced in Section 3.2. Define a Lyapunov function for (2) to be a positive-definite function $V(x, t)$ with $\dot{V}(x, t) = \frac{\partial V(x,t)}{\partial x}\hat{f}(x, t) < 0$. For some $t \in [0, T]$ and $\rho : [0, T] \to \mathbb{R}_{\geq 0}$, define the *$\rho$-sub-level set* of $V(x, t)$ as

$$\mathcal{L}(t) = \{x \mid V(x, t) \leq \rho(t)\}.$$

Let $X_i, X_j \subset \mathbb{R}^n$ be regions in the configuration space where the regions are adjacent. Consider a trajectory ξ whose initial initial value is $\xi(0) \in X_i$ and final value is $\xi(T) \in X_j$ and at every time instant stays in $X_i \cup X_j$. The problem of computing an *invariant funnel* is one where we verify the largest set of states about ξ that satisfy the same properties as ξ; namely, reachability from X_i to X_j and invariance to $X_i \cup X_j$. Such a set is found via the following maximization

problem:

$$\max_{\rho(t)\geq 0} \rho(t), \qquad t \in [0,T] \tag{3}$$

$$\text{s.t.} \quad V(x,t) \leq \rho(t) \implies \dot{V}(x,t) \leq \dot{\rho}(t), \quad \forall t \in [0,T], \tag{4}$$

$$\mathcal{L}_{ij}(T) = \{x \mid V(x,T) \leq \rho(T)\} \subseteq G, \tag{5}$$

for a goal set $G \subseteq X_j$. Computational tools such as the semidefinite programming solvers SeDUMI and MOSEK can solve a time-discretized form of the above problem, when expressed as a sums-of-squares program. Such tools apply to systems that are expressed as differential equations that are polynomial in their arguments; for this reason, we assume system models of this form throughout this paper. In [9], additional constraints are introduced to compute *bounded funnels* that are guaranteed to stay within an invariant $X_i \cup X_j$,

$$\mathcal{L}_{ij}(t) = \{x \mid V(x,t) \leq \rho(t)\} \subseteq X_i \cup X_j, \tag{6}$$

for all $t \in [0,T]$. We denote any bounded funnel $\mathcal{L}_{ij}(t)$ as a *transition funnel* if $i \neq j$ and any bounded funnel $\mathcal{L}_i(t) = \mathcal{L}_{ii}(t)$ as an *inward-facing funnel*.

We briefly summarize the algorithm in [9], which provides a twofold procedure for constructing a library of controllers for all transitions in a state machine $\mathcal{A}$. First, let I_i be the index set of successors to state s_i; i.e. $I_i = \{k \mid \exists \sigma \in 2^{AP_e} : (s_k, \sigma, s_i) \in \delta\}$. For a particular $s_i \in S$ and $j \in I_i$, construct a controller and label the associated funnels $\mathcal{L}_{ij}$.

Next, if I_i contains two or more elements, additional controllers must be created to fulfill *reactive behaviors* of $\mathcal{A}$; that is, behaviors that ensure, for all time instants along any trajectory in $\mathcal{L}_{ij}$ enabling the transition from X_i to X_j, the system is able to react to sensor inputs and move instead to X_k, $k \in I_i$, $k \neq j$. In this case, denote the *reactive-composition* (RC) set as the set in which any outgoing transition from the current region to successor regions is possible, i.e. $\bigcap_{k\in I_i}[\mathcal{L}_{ik}]$. For each $\mathcal{L}_{ij}$ and $\mathcal{L}_i$, compute a new inward-facing funnel $\mathcal{L}_i$ whose initial set is taken from $\mathcal{L}_{ij}$ such that $[\mathcal{L}_{ij}] \cap \partial X_i \subseteq [\mathcal{L}_i]$ and whose final set $G = \bigcap_{j\in I_i}[\mathcal{L}_{ij}]$, each time storing them into the library. The composition requirement $[\mathcal{L}_{ij}] \cap \partial X_i \subseteq [\mathcal{L}_i]$ assures that the system can react to the environment regardless of which trajectory is used in $\mathcal{L}_{ij}$, up to the time at which it leaves X_i. More precisely, for all times t for which $\xi(t) \in [\mathcal{L}_{ij}] \cap X_i$, there exists a time $t' \geq t$ for which $\xi(t') \in [\mathcal{L}_i]$.

For example, consider a unicycle model consisting of three continuous states (x, y, θ): two Cartesian displacements and an orientation angle and whose command input is its angular rate, ω. The unicycle is assumed able to turn but with turning radius limited to remain below a fixed threshold and may only move in the forward direction with fixed velocity. For the example shown in Fig. 3, a controller and funnel $\mathcal{L}_{12}$ is constructed to implement movement between regions X_1 to X_2 using an under-actuated unicycle model. With $\mathcal{L}_{12}$, another controller and funnel $\mathcal{L}_1$ is computed to implement the transition keeping the system in X_1. The intersection of $[\mathcal{L}_{12}]$ and $[\mathcal{L}_1]$ (the RC set) indicates the states in which both transitions $(s_1, \texttt{True}, s_2)$ and $(s_1, \texttt{False}, s_1)$ can be achieved. Note that the maximization in (3) helps us to achieve a maximal set with respect to the given trajectory. Finally, the goal set G is determined from composition requirements. In particular, the set G when computing $\mathcal{L}_{12}$ is X_2, while G for $\mathcal{L}_1$ is the computed funnel $\mathcal{L}_{12}$. For the remainder of this paper, we assume funnels are computed in this manner.

Note that that above algorithm can fail at finding such an inward-facing funnel $\mathcal{L}_i$ in certain cases. For example, consider the unconstrained funnel shown in Fig. 2b. The funnel shown satisfies the reactivity requirement $[\mathcal{L}_{12}] \cap \partial X_1 \subseteq [\mathcal{L}_1]$, but not the invariance condition $[\mathcal{L}_1] \subseteq X_1$. If the funnel were smaller, it could satisfy invariance, but at the expense of reactivity for all trajectories in $\mathcal{L}_{12}$. The proposed approach outlined in the next section directly addresses this issue by finding a controller and funnel for a given trajectory that satisfies both the reactivity condition and the invariance condition.

4. PROBLEM STATEMENT

In this paper, we address the following two problems.

Problem 1 (Fulfillment of a reactive behavior for $s_i \in S$)**.** *Given the dynamics f and g, a transition funnel $\mathcal{L}_{ij}$, region X_i, a goal set $G \subseteq X_i$, find a trajectory ξ, μ where $\xi(t) \in X_i$ for all $t \in [0,T]$ and synthesize a low-level controller $u(t)$ and a funnel $\mathcal{B}(t) \subseteq X_i$ such that $[\mathcal{L}_{ij}] \cap X_i \subseteq [\mathcal{B}]$.*

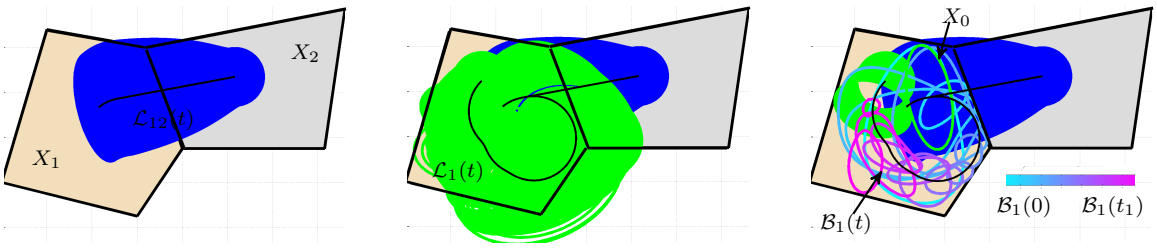
Problem 2 (Partial fulfillment of a reactive behavior for $s_i \in S$, re-partitioning the workspace)**.** *If Problem 1 fails to be solved, given the dynamics f and g, a transition funnel $\mathcal{L}_{ij}$, region X_i, a goal set $G \subseteq X_i$, find a trajectory ξ, μ where $\xi(t) \in X_i$ for all $t \in [0,T]$ and synthesize a low-level controller $u(t)$ and a funnel $\mathcal{B}(t) \subseteq X_i$. Compute a new region X_{ij} within which the system cannot react to the environment when executing a controller $\mathcal{L}_{ij}$, and accordingly re-write the specification.*

Note that Problem 1 does limit the discussion to inward-facing funnels. We may find *transition* funnels for $(s_\ell, \cdot, s_i)$ by setting a goal $G \subseteq X_j$, replacing the invariant X_i with $X_i \cup X_j$, and relaxing the composition requirement to $\mathcal{L}_{\ell i}(T) \subseteq [\mathcal{B}]$, provided we are given $\mathcal{L}_{\ell i}$ such that $\mathcal{L}_{\ell i}(T) \subseteq X_i$.

5. CONFORMING FUNNELS

In this section, we introduce an approach that attempts to compute a bounded funnel when the procedure outlined in Section 3.3 fails to do so. We do this by making use of the notion of control barrier functions; functions whose parameters can be tailored to achieve a certificate of safety with respect to an unsafe set. Such functions parameterize a nonlinear feedback control law that is executed by the system for the certificate to hold. In our case, for a given trajectory, we generate a sequence of certificates and feedback controllers for the system, giving rise to a funnel that conforms to the required invariant set.

In more precise terms, given trajectories ξ, μ, and controller K, the problem is to find controller parameters that verify the largest set of states that satisfies the invariance and reachability properties of ξ with respect to an invariant set. The controller leverages the proven idea of unified control barrier functions and control Lyapunov functions, which is extended in this section to the case of time-varying trajectory stabilization. While the developments in this section are largely inspired by the work of [25, 21], they are complementary to those works as we also offer a computational framework through sums-of-squares optimization to numerically compute such barrier functions and controllers.



(a) (b) (c)

Figure 2: Constructing conforming funnels for a state machine: $S = \{s_1, s_2\}$ with transitions $\{(s_1, \texttt{True}, s_2), (s_1, \texttt{False}, s_1)\}$. (a) shows a funnel $\mathcal{L}_{12}$ computed for a three-state unicycle model (the funnel shown is a projection onto $\mathbb{R}^2$) for a low-level controller satisfying the transition. (b) shows a funnel $\mathcal{L}_1$ (green) that violates the invariant preventing it from entering X_2. (c) shows a conforming funnel $\mathcal{B}_1(t)$ that satisfies the transition of staying in X_1. $\mathcal{B}_1(t)$ consists of a segment of $\mathcal{L}_1$ (green), and a barrier whose zero-level set is shown at specific time instants.

5.1 Using Control Barrier Functions to Satisfy State Constraints

Suppose we extend the control input as $u(t) = K(t)(x(t) - \xi(t)) - \mu(t) + \bar{u}(t) = U(t) + \bar{u}(t)$. We then obtain the following *augmented* system

$$\dot{x} = \hat{f}(x,t) + g(x)\bar{u}, \qquad t \in [0,T]. \tag{7}$$

The authors of [25] show that there exists a control $\bar{u}(x)$ yielding a control barrier function (CBF) that ensures safety of trajectories within a nonlinear system. In this case, the CBF provides a certificate of *safety*, but in general does not guarantee convergence, which we require in order to ensure reachability of trajectories from an initial set to a goal set. Theorem 3 in [21] remedies this by expressing a set of conditions to construct a unified CBF and control Lyapunov function (CLF) to ensure safety of a time-invariant system whose static equilibrium lies in the interior of the barrier.

In this work, we extend this to the time-varying case for systems of the form of (7), where the problem is to find a controller that stabilizes to a *trajectory* rather than a static equilibrium.

Consider a function $B(x,t)$ defined over an interval $t \in [0,T]$ and let $a(x,t) = \frac{\partial B(x,t)}{\partial x} f(x,t) + \frac{\partial B(x,t)}{\partial t}$ and $b^T(x) = \frac{\partial B(x,t)}{\partial x} g(x)$. The conditions for the time-varying problem assert that, if $B(x,t)$ can be found that satisfies the following conditions:

$$x \in X_u \implies B(x,t) \geq 0 \tag{8}$$

$$\frac{\partial B(x,t)}{\partial x} g(x) = 0 \implies \frac{\partial B(x,t)}{\partial x} \hat{f}(x,t) + \frac{\partial B(x,t)}{\partial t} < 0 \tag{9}$$

$$x \in X_0 \implies B(x,t) \leq 0 \tag{10}$$

then the control law:

$$\bar{u} = \begin{cases} -\frac{a + \sqrt{a^2 + \gamma^2 b^T b}}{b^T b} b \text{ for } b \neq 0 \\ 0 \text{ for } b = 0 \end{cases} \tag{11}$$

ensures the safety of the system with respect to the initial conditions X_0, where $\gamma > 0$ is a tunable parameter. If found, $B(x,t)$ is a *time-varying control barrier function* (TV-CBF). In words, for a given $t \in [0, t_1]$, with $0 < t_1 \leq T$, the level set $B(x,t) = 0$ certifies a safety bound where $B(x,t)$ is strictly positive inside the unsafe set X_u, and strictly negative within the initial set X_0, provided that the control input $\bar{u}$ is applied for all x on the barrier certificate (i.e. in the set $\{x \mid B(x,t) = 0\}$).

In similar fashion to [21], we also show that the control law $\bar{u}$ (namely (11)), when applied in the interior of the barrier ($B(\xi(t),t) \leq 0$) simultaneously stabilizes the system to the trajectory and keeps the system safe within the barrier. The following provides a statement on the safety and reachability properties if the above conditions hold true:

Proposition 1. *Given the trajectory-stabilizing feedback system* (2)*, an unsafe set X_u, and some initial set $X_0 \subset \mathbb{R}^n \backslash X_u$, then a feasible solution to* (8)–(10) *yields a safe trajectory with respect to X_u and guarantees trajectory convergence if $u(t) = U(t) + \bar{u}(t)$, with $\bar{u}(t)$ satisfying* (11)*.*

Proof. To show that $\bar{u}(t)$ satisfying (11) gives rise to a trajectory stabilizing controller follows by showing that $B(x,t)$ is smooth (as per Lemma 3.6 in [11]), and that it results in a system that is a control Lyapunov function. Smoothness follows trivially by construction of $B(x,t)$. When $\frac{\partial B(x,t)}{\partial x} g(x) \neq 0$, then

$$\frac{\partial B(x,t)}{\partial x} \hat{f}(x,t) + \frac{\partial B(x,t)}{x} g(x)\bar{u}(t) = -\sqrt{\left\| \frac{\partial B(x,t)}{x} \hat{f}(x,t) \right\|^2 + \gamma \left\| \frac{\partial B(x,t)}{x} g(x) \right\|^2} < 0.$$

for $\gamma > 0$ and $t \in [0, t_1]$. Otherwise (when $\frac{\partial B(x,t)}{\partial x} g(x) = 0$), $\frac{\partial B(x,t)}{\partial x} \hat{f}(x,t) + \frac{\partial B(x,t)}{\partial t} < 0$ holds, by construction, from condition (9). As $u(t) = U(t) + \bar{u}(t)$ is applied whenever $B(x,t) \leq 0$, and, for a given $t \in [0,T]$, then $B(x,t) = \min_x B(x,t)$ precisely when $x = \xi(t)$, the system is attractive to the trajectory.

We must also show that such controllers guarantee invariance to the barrier. If (8)–(10) is feasible for X_0 then, by construction, there exists a $B(x,t)$ for which $B(x,t) = 0$ separates X_u and X_0 for any given $t \in [0, t_1]$. □

Notice that the problem of finding the TV-CBF in (8)–(10) is within the sums-of-squares class of problems for cases where the system and constraint sets are polynomial in x and t. We have encoded this as such and adopt the package MOSEK in this work as the solver. Note also that the problem of trajectory stabilization generalizes many aspects of robotics in the case where there is no single global equilibrium; for example, the under-actuated unicycle model introduced in the previous section. For problems where it

is possible to find a sufficiently large region of attraction about a static equilibrium, the time-invariant counterpart of Proposition 1 will suffice (Theorem 3 of [21]).

To summarize the result, assume we have an augmented system of the form (7) that is stable about a neighborhood of a nominal trajectory ξ, μ over a finite time horizon $t \in [0, T]$ (recall that this is true because u subsumes a time-varying LQR controller). Solve for a barrier function that meets state constraints (required invariants for the funnel) according to (8)–(10). If one is found, then this barrier function parameterizes a control law of the form (11) that, when applied to the augmented system, ensures that, for all states starting in the set $\{B(x,t) \leq 0\}$, the system is guaranteed to converge to the trajectory (make progress along the trajectory) and remain invariant to the zero-level set of $B(x,t)$ for all $t \in [0, T]$. The resulting set $\mathcal{B}(t) : \mathbb{R}_{\geq 0} \to \{x \mid B(x,t) \leq 0\}$ is a conforming funnel for the feedback-controlled system.

5.2 Computational Approach

As opposed to the funnel formulation of Section 3.3 in which the objective is to search for a $\rho(t)$ for a function with fixed parameterization, the TV-CBF conditions search directly for a barrier that is a function of x. This generally requires a larger number of decision variables, giving rise to a greater number of computations. We therefore outline a procedure for computing funnels using the TV-CBF objectives with small computational expense by bootstrapping it with an unconstrained funnel.

The procedure is as follows. As pictured in Fig. 3a–3b, compute first an unconstrained funnel according to the conditions (3) (i.e. without the addition of (6)). If this happens to satisfy the constraints (6), then we accept this funnel and return. Otherwise, as illustrated in Fig. 3c, select t_1 as the minimum time where $X_i \cup X_j$ contains the portion of $\mathcal{L}(t)$ for $t \in [t_1, T]$. On the interval $(t_1, T]$, we let $\mathcal{L}_{suff} = \{\mathcal{L}(t) \mid t \in (t_1, T]\}$ be a *suffix* of the original funnel that is safe with respect to the invariants over for all $t \in (t_1, T]$. We then set the initial condition as $X_0 = \mathcal{L}_{ij}(t^*)$ (where t^* will be introduced in Sec. 6.1). In order to enforce composability of $\mathcal{B}(t)$ with the suffix funnel, we ensure that $\mathcal{B}$ is contained inside $\mathcal{L}$ at time t_1. Therefore, at time t_1, we apply $\mathcal{L}(t_1)$ is designated as the unsafe set X_u, while, at all other times $t = [0, t_1)$, we set $X_u = \mathbb{R}^n \backslash X_i$. We also set the constraint that $[\mathcal{L}_{ij}] \cap X_i \implies B(x,t) = 0$. This constraint ensures that the generated funnel is valid with respect to the reactivity condition for all trajectories in $\mathcal{L}_{ij}$.

We then proceed to compute a $\mathcal{B}(t)$ according to (8) as per Proposition 1 over the interval $[0, t_1]$. The computed conforming funnel and suffix funnel combine to form an aggregate funnel that verifies reachability to the goal set G and safety with respect to X_u for a well-defined neighborhood about the trajectory:

$$\mathcal{B}(t) = \begin{cases} \{x \mid B(x,t) = 0\} & \text{for } 0 \leq t \leq t_1 \\ \mathcal{L}_{suff}(t) & \text{for } t_1 < t \leq T \end{cases}$$

Note that $\mathcal{B}(t)$ is merely $\mathcal{L}(t)$ for the case where the TV-CBF computations are not necessary.

Observe that we do not preserve the portion of the funnel *before* it leaves the invariant set because the barrier computations require an over-approximation to the initial set in order to guarantee composition of funnels. At such time instants, this imposes tight constraints on the resulting barrier (the barrier would have to be larger than the last non-red ellipse shown in Fig. 3b, but smaller than the boundary). In practice, this has rendered the optimization problem infeasible in the majority of cases tested.

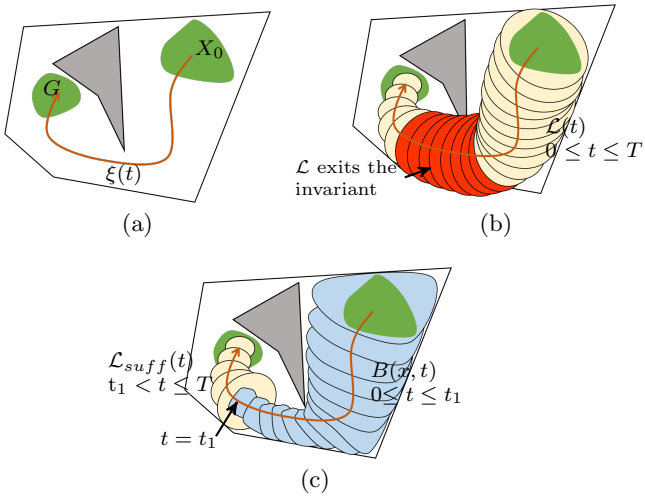


Figure 3: Generating conforming funnels. (a) A trajectory is created steering the system from the start set X_0 to G. (b) A fixed-parameter funnel is created, and checked for collisions. (c) $\mathcal{L}$ exits the invariant up to $t = t_1$; therefore a conforming funnel is computed respecting the invariant up to t_1.

6. WORKSPACE RE-PARTITIONING AND ABSTRACTION GENERATION

In the case where conforming funnels cannot be created for the high-level behavior (for the choice of trajectory), we may use controllers that do not verify the behavior exactly to create new workspace partitions, and update the discrete abstraction of the system on those partitions. In this vein, we introduce the second main contribution of this paper: an algorithm for constructing controllers and partitions in the case that a controller has not been found to satisfy a particular high-level behavior. We furthermore introduce a procedure for updating a specification written as linear temporal logic based on the results of the partitioning. The approach to partitioning lends well to automatic generation of intuitive explanations to users in the case where no satisfying finite state machine can be found; we discuss these implications via a case study.

6.1 Partial Fulfillment of High-Level Behaviors

Our goal is to find inward funnels for the entirety of the time the system spends in X_i. Notice that if, for a particular transition funnel $\mathcal{L}_{ij}$, a funnel $\mathcal{B}_i$ is found according to the criteria set forth in the Sec. 5, the system is reactive all the way up to the boundary; i.e. $[\mathcal{L}_{ij}] \cap \partial X_i \subseteq [\mathcal{B}_i]$. This means the transition $(s_i, \cdot, s_j)$ in $\mathcal{A}$ has been implemented exactly.

Otherwise, there exists a gap between the inward funnel and the region boundary where the robot may no longer reach the RC set without first crossing the boundary. We call such a gap a *reactivity gap* which is nonzero whenever there exist trajectories in $\mathcal{L}_{ij}$ that do not intersect

with the set $\mathcal{B}_i$ up to the point at which it leaves X_i, i.e. $[\mathcal{L}_{ij}] \cap \partial X_i \not\subseteq [\mathcal{B}_i]$. Strictly speaking, this would result in failure to generate low-level controllers that satisfy $\mathcal{A}$, or else necessitate searching for new transition funnels $\mathcal{L}_{ij}$ that result in zero reactivity gap, a potentially expensive process with no termination guarantees.

In order to proceed with generating controllers in the case where such a transition cannot be fulfilled, we modify the original specification via a re-partitioned workspace given the existing set of low-level controllers. We do so with an approach that minimizes the reactivity gap modulo the given $\mathcal{L}_{ij}$. This translates into the problem of searching for a trajectory and funnel that is able to guarantee reactive composition and invariance to the region as late as possible along the transition funnel $\mathcal{L}_{ij}$; i.e. find a $\mathcal{B}_i^*$ as the argument for which

$$t^* = \max_{\mathcal{B}_i \text{ s.t. } (8)\text{–}(10)} \{t \mid \mathcal{L}_{ij}(t) \cap X_i \subset [\mathcal{B}_i]\}. \tag{12}$$

Given $\mathcal{B}_i^*$, the reactivity gap for $\mathcal{L}_{ij}$ is defined as a new region

$$X_{ij} = \{\mathcal{L}_{ij}(t) \mid t \in [t^*, T]\} \backslash [\mathcal{B}_i^*] \cap X_i. \tag{13}$$

For visualization purposes, we can over-approximate X_{ij} via a polytope projection onto, for example, a 2-D workspace map. With this new region defined, a procedure for updating the discrete abstraction in the context of linear temporal logic specifications is provided in the next section.

6.2 Robot Abstractions for Linear Temporal Logic Specifications

Reactive tasks may be expressed as linear temporal logic (LTL) specifications. LTL formulas are defined over the set AP of atomic (Boolean) propositions by the recursive grammar $\varphi ::= \pi \in AP \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$. The following operators are derived from the Boolean operators $\wedge$ "conjunction" and $\neg$ "negation", and the temporal operators $\bigcirc$ "next" and $\mathcal{U}$ "until": "disjunction" $\vee$, "implication" $\Rightarrow$, "equivalence" $\Leftrightarrow$, "always" $\Box$, and "eventually" $\Diamond$. For a description of the semantics of LTL, we refer the reader to [24].

Letting $\pi \in AP_e \cup AP_s$ represent a binary proposition, a mission specification is expressed as an LTL formula of the form:

$$\varphi := \varphi^e \implies \varphi^s,$$

where φ^e and φ^s are defined over $AP_e \cup AP_s \cup \bigcirc AP_e \cup \bigcirc AP_s$, and are further decomposed into formulas for initial conditions, safety conditions to be satisfied always, and goals to be satisfied infinitely often. Considering specifications of this form allows us to take advantage of efficient algorithms for synthesizing a finite-state machine $\mathcal{A}$ that satisfying the specification [4].

In the context of LTL mission specifications, a *discrete abstraction* is a set of LTL formulas that model the discrete behavior of the underlying continuous system. We encode discrete abstractions using the LTL encoding of [10], which takes into account actions with arbitrary completion times, allowing for a rich set of behaviors to occur concurrently. Under this encoding, let $\pi_{a\ell} \in AP_s$ denote the proposition that is `True` when activating a transition to some region X_ℓ, and $\pi_\ell \in AP_e$ denote the proposition that is `True` when motion to X_ℓ has been completed.

If $\mathcal{B}_i^*$ is computed such that there is a nonempty reactivity gap, i.e. $[\mathcal{L}_{ij}] \cap \partial X_i \not\subseteq [\mathcal{B}_i^*]$, we update the abstraction as follows. The first step is to compute a reactivity gap, as the set according to (13), then extend the set of propositions AP_e with a new region proposition π_{ij} that is `True` when the system is in X_{ij}. We then append the following formula to the formula for environment assumptions φ^e:

$$\begin{aligned}
&\Box\left(\pi_{ai} \wedge \pi_i \wedge \bigcirc \pi_i \wedge \neg\pi_{ij} \implies \bigcirc\neg\pi_{ij}\right) \wedge \\
&\Box\left(\pi_{aj} \wedge \pi_i \wedge \neg\pi_{ij} \implies \bigcirc \pi_i\right) \wedge \\
&\Box\left(\pi_{aj} \wedge \pi_i \wedge \pi_{ij} \implies \bigcirc\pi_{ij} \vee \bigcirc \pi_j\right) \wedge \\
&\Box\left(\bigcirc\neg\pi_i \implies \bigcirc\neg\pi_{ij}\right).
\end{aligned} \tag{14}$$

The first condition enforces that, if the system is commanded to remain in X_i, it must not enter X_{ij}. The second and third conditions enforce that if the robot is commanded to enter X_j when in X_i, it cannot do so without first entering X_{ij}. The last condition encodes the fact that X_{ij} is a subset of X_i.

6.3 Putting It All Together: Low-Level Controller Synthesis

We now describe an algorithm for constructing verified atomic controllers satisfying the reactive composition property according to criterion (12) for a transition from a state i to another state j. In line 8 of Algorithm 1, the optimization problem (8)–(10) is solved with the additional criteria where the resulting barrier conforms *precisely* for the portion of the invariant that intersects with the transition $\mathcal{L}_{ij}$. If this problem has a solution, this means that for any controlled trajectory from X_i to X_j, the system is verified to be able to react to the environment and move instead to any X_k, $k \in I_i$, $k \neq j$. For instance, if the robot may be represented by a fully-actuated (e.g. holonomic) model where the robot can move instantaneously in any direction, then the algorithm will be able to create controllers that satisfy a transition that exploits this property to its fullest. If this does not succeed, then line 10 solves the optimization again without this restriction. If successful, the abstraction is updated in line 12, and the funnel is returned in line 22. Otherwise, the time t_{init} is reduced in line 19 and the process repeats. Note that line 2 is a restatement of (12) that finds a suboptimal t^*, t_{init}, by iterating backward in time a fixed amount τ.

Upon returning from the algorithm, a decision is made to synthesize another finite state machine (if necessary with the aid of any LTL-based revisions as explained in [8]). If the abstraction has changed after constructing controllers for each transition, then synthesis takes place; otherwise it does not. If this new finite state machine is simulated by the old one, i.e. the new one contains a subset of the behaviors of the old one, then the existing library of low-level controllers is deemed sufficient for the finite state machine. Otherwise, new low-level controllers are generated in a manner that reuses as much of the existing library as possible.

7. EXAMPLE: A BOX TRANSPORTATION TASK

In this human-interactive problem domain, the task is to move packages from a pick-up area to one of two drop-off locations, as pictured in Fig. 1. Formally, "Visit the loading area. If push_box is active and go_to_left is requested, visit

Algorithm 1 Computing an inward-facing funnel $\mathcal{B}_i$ for s_i to implement a change from successor state s_j to s_k, $j, k \in I_i, j \neq k$.

procedure REACTIVECOMPOSITION($\mathcal{L}_{ij}$,X_i,X_j,τ)
  $t_{init} \leftarrow \arg\max_t\{t \mid \mathcal{L}_{ij}(t) \cap X_i \neq \emptyset\}$
  while $t_{init} \geq 0$ and not feasible **do**
    create a trajectory from $\mathcal{L}_{ij}(t_{init})$ to the RC set for state i
    $\mathcal{L}_i$, feasible $\leftarrow$ solution to (3)
    if feasible and $[\mathcal{L}_i] \not\subseteq X_i$ **then**
      $t_1 \leftarrow \arg\max_t\{t \mid \mathcal{L}_i(t) \cap (\mathbb{R}^n \backslash X_i) \neq \emptyset\}$
      $\mathcal{B}_i$, feasible $\leftarrow$ solution to (8)–(10) with additional constraints $[\mathcal{L}_{ij}] \cap X_i \implies B_i(x,t) = 0$
      if not feasible **then**
        $\mathcal{B}_i$, feasible $\leftarrow$ solution to (8)–(10)
        if feasible **then**
          compute X_{ij} and update the abstraction as (14)
        end if
      end if
    else if feasible **then**
      $\mathcal{B}_i \leftarrow \mathcal{L}_i$
    end if
    if not feasible **then**
      $t_{init} = t_{init} - \tau$
    end if
  end while
  return $\mathcal{B}_i$
end procedure

dropoff_L. If push_box is active and go_to_right is requested, visit dropoff_R. Activate push_box when in pickup and deactivate push_box when in dropoff_L or dropoff_R." We employ a KUKA youBot to perform the task, which operates on an omnidirectional base whose position and orientation is measured in real time. Packages are moved by way of pushing them along the ground using the robot's front fender. There is one action (treated as a system variable) in this scenario, push_box, which is `True` whenever the robot is moving a package and `False` otherwise.

In our discrete abstraction of the problem, we do not explicitly model the dynamics of the box, but rather we impose certain conditions that are required of the dynamics in order to assure that the box always maintains contact with the robot. We therefore impose the dynamics of a *unicycle* that is capable of forward velocities up to a certain forward velocity and within a given range of turning radii. Suitable maximum linear and angular velocities were determined experimentally in order to ensure the box is always in contact with the robot. In order to disengage the box, we impose holonomic (fully-actuated) dynamics with negative forward velocity. When push_box is `False`, the holonomic model is put into effect; otherwise, the unicycle model is used. We use the proposed approach to design controllers for this example.

7.1 Synthesis and Workspace Re-Partitioning

We express the task as a mission specification (linear temporal logic formulas), from which we synthesized a high-level state machine using the slugs synthesis tool[1]. Our state machine consists of 6 states and 9 transitions. Controllers were synthesized using the proposed conforming funnels approach and Algorithm 1. Our approach was coded as a set of MATLAB routines; with our implementation, each reactive funnel took on average 5.2 minutes to compute on a laptop with Intel Core i7 2.8GHz processor and 8GB of RAM. The original specification and dynamics necessitated an update to the abstraction that, in turn, required an update to the original specification.

[1]http://github.com/LTLMoP/slugs

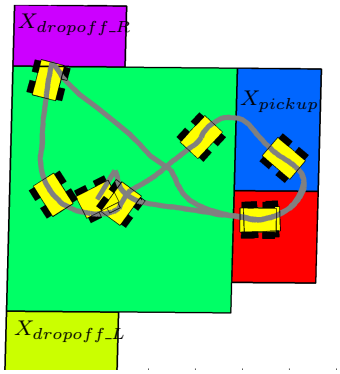


Figure 4: A trajectory for the box-transport example for the case where the environment is continually issuing the request go_to_right.

Based on the abstraction update, a new finite state machine could not be synthesized without additional environment assumptions. Using the revisions approach algorithm in [8], we automatically generated environment assumptions that restrict its behavior in the newly-generated reactive gap regions. Specifically, given the regions $X_{middle,L}$ and $X_{middle,R}$, additional assumptions were generated in the form of LTL formulas added to the environment subformula. In natural language, the formulas state the following: "If the robot is in region middle,L, heading to dropoff_L and activating push_box do not issue the request go_to_right," and "If the robot is in middle,L, heading to dropoff_R and activating push_box do not issue the request go_to_left." With this new controller (10 states and 9 transitions), the low-level controller synthesis process continued for the new state machine.

7.2 Execution of the Conforming Funnels

The proposed approach was evaluated in a set of laboratory experiments. The hybrid controller was executed on a laptop computer connected wirelessly to the youBot using the Robot Operating System (ROS) as the communication layer. The controller ran at a rate of approximately 5 Hz and the robot pose was obtained via a motion capture system.

A trajectory showing the execution of the controller on the youBot is pictured in Fig. 4 for the case where the environment is static. Notice that, when push_box is `True`, the robot executes the unicycle model and hence makes gradual turns; otherwise the holonomic model is active and the robot follows straight paths. An example of the robot reacting to the environment is depicted in Fig. 5. In Fig. 5c, the robot is executing a conforming funnel in order to fulfill a changing request just before the robot enters dropoff_L. The robot's resulting motion is smooth, remains within the funnel, and can be seen to converge to the nominal trajectory. A complete video demonstrating the controllers is available at: `https://youtu.be/1XEV0Ga3UEY`.

The conforming funnels approach was compared against funnels generated using the approach from [9]. As can be

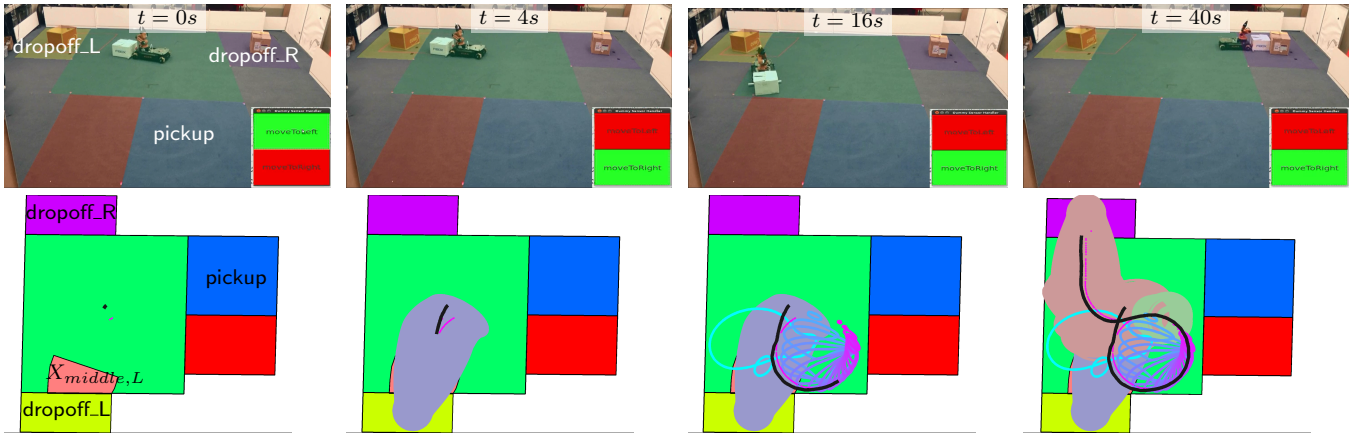


Figure 5: Experimental results for the box-transportation scenario under a dynamic environment, a sequence of images captures taken from the video available at: `https://youtu.be/1XEV0Ga3UEY`. (a)–(d), top, show the robot at several time instants as it delivers a box, with the inset showing the sensor values go_to_left, go_to_right. The map is diplayed at bottom of each subfigure, along with the robot's trajectory (black), the activated funnels, and the nominal trajectory for each funnel (magenta). The pink region in (a) is the reactivity gap region $X_{middle,L}$. Note that the sensor value changes from go_to_left to go_to_right at the time instant shown in (b).

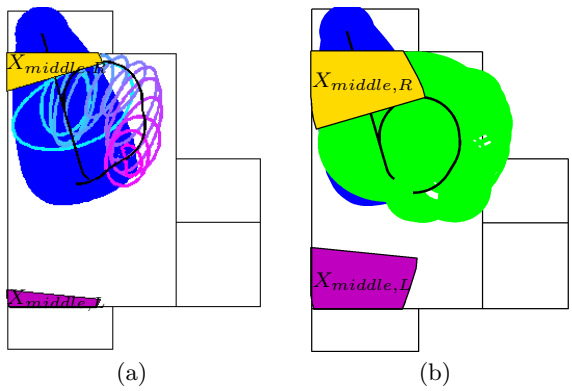


Figure 6: Evaluation of the reactivity of the system to changes in the environment for the box-transport problem. (a) and (b) show reactivity gap regions computed using, respectively, the conforming funnels approach and funnels approach from [9].

seen in Fig. 6a, the reactivity gap regions generated with the proposed approach are considerably smaller than those computed in the formulation of [9] in Fig. 6b. This is due to the fact that such a conforming funnel can be computed using a trajectory starting close to the drop-off region and inside the transition funnel yet still also able to verify the set of states in the transition funnel. The implication of this improvement is a greater responsiveness to the environment when the robot is in close proximity to the drop-off areas.

8. CONCLUSIONS

In this paper, we address two problems: (1) synthesizing hybrid controllers that allow a robot to react to its environment whenever dynamically feasible given the geometry of the workspace, and (2) an automated approach for transforming dynamically-informative revisions into changes to the specification, in the case that a controller cannot be obtained for a high-level behavior. To address the first, we contribute a method for synthesizing correct-by-construction controllers with respect to given high-level behaviors, the geometrical constraints of the workspace, and a robot dynamics model. Our approach generalizes existing techniques for nonlinear system verification using fixed-parameter funnels. In situations where the high-level behaviors cannot be fulfilled strictly, we also contribute an algorithm that uses the controllers to re-partition the workspace and automatically adapt the high-level specification with a new discrete abstraction generated on these new partitions. The approach is general; it is intended for a finite state machine synthesized from any reactive specification and any system model provided it can be expressed in terms of polynomial ODEs.

In the future, we will extend the approach to be more complete so that controllers can be generated over a wider portion of the state space. To address this challenge, it will become necessary to explore the possibility of optimizing among trajectories that both fulfill the high-level behavior and result in funnels that are maximal. Future work also includes exploring the significance of parameter adaptation in the construction of TV-CBF-based controllers; for example, the order of the polynomials used to parameterize the barrier function. Using the TV-CBF method, we intend to explore the computation of robust motion primitives that admit controllers that are provably correct over a wide array of parameters for the system model. Lastly, we intend to use these primitives as an aid for performing synthesis when unknown changes to the system model occur during execution.

Acknowledgement

This work was supported by NSF Expeditions in Computer Augmented Program Engineering (ExCAPE). The authors thank Prof. Ross Knepper and Prof. Alex Vladimirsky for valuable discussions leading to the formation of this work.

9. REFERENCES

[1] R. Alur, R. Bodík, G. Juniwal, M. M. K. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 1–8, 2013.

[2] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA 2010)*, pages 2689–2696, 2010.

[3] A. Bhatia, M. Maly, L. Kavraki, and M. Vardi. Motion planning with complex goals. *Robotics Automation Magazine, IEEE*, 18(3):55 –64, sept. 2011.

[4] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[5] R. Burridge, A. Rizzi, and D. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18:534–555, 1999.

[6] E. Clarke. Counterexample-guided abstraction refinement. In *10th International Symposium on Temporal Representation and Reasoning / 4th International Conference on Temporal Logic (TIME-ICTL 2003)*, page 7, 2003.

[7] D. C. Conner, A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Proc. of 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2003)*, volume 4, pages 3546– 3551, 2003.

[8] J. A. DeCastro, R. Ehlers, M. Rungger, A. Balkan, P. Tabuada, and H. Kress-Gazit. Dynamics-based reactive synthesis and automated revisions for high-level robot control. *CoRR*, abs/1410.6375, 2014.

[9] J. A. DeCastro and H. Kress-Gazit. Synthesis of nonlinear continuous controllers for verifiably-correct high-level, reactive behaviors. *The International Journal of Robotics Research*, 34(3):378–394, 2015.

[10] J. A. DeCastro, V. Raman, and H. Kress-Gazit. Dynamics-driven adaptive abstraction for reactive high-level mission and motion planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA 2015)*, pages 369 – 376, Seattle, WA, 2015.

[11] Z. Jiang, Y. Lin, and Y. Wang. Stabilization of nonlinear time-varying systems: a control lyapunov function approach. *J. Systems Science & Complexity*, 22(4):683–696, 2009.

[12] D. Kirk. *Optimal Control Theory: An Introduction.* Prentice-Hall, 1976.

[13] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from ltl specifications. In *Proc. of the 9th Int. Conf. on Hybrid Systems: Computation and Control (HSCC'06)*, pages 333–347, Berlin, Heidelberg, 2006. Springer-Verlag.

[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal logic based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[15] J. Liu and N. Ozay. Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *Proc. of the 17th Int. Conf. on Hybrid Systems: Computation and Control (HSCC'14)*, 2014.

[16] J. Liu, U. Topcu, N. Ozay, and R. M. Murray. Reactive controllers for differentially flat systems with temporal logic constraints. In *Proc. of the 51st IEEE Conf. on Decision and Control (CDC 2012)*, pages 7664–7670, 2012.

[17] J. Maidens and M. Arcak. Trajectory-based reachability analysis of switched nonlinear systems using matrix measures. In *Prof. of the 53rd IEEE Conf. on Decision and Control (CDC)*, pages 6358–6364, Dec 2014.

[18] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *Proc. of the 16th Int. Conf. on Hybrid Systems: Computation and Control (HSCC'13)*, pages 353–362, Philadelphia, PA, 2013.

[19] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Contr.*, 50(7):947–957, 2005.

[20] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proc. of the 4th Int. Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, pages 477–492, 2004.

[21] M. Romdlony and B. Jayawardhana. Uniting control lyapunov and control barrier functions. In *Prof. of the 53rd IEEE Conf. on Decision and Control (CDC)*, pages 2293–2298, Dec 2014.

[22] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *the International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[23] C. J. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of IEEE*, 88:949–969, July 2000.

[24] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for concurrency*, pages 238–266. Springer, 1996.

[25] P. Wieland and F. Allgower. Constructive safety using control barrier functions. In *Proc. of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 473–478, 2007.

[26] E. Wolff, U. Topcu, and R. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2013)*, pages 4332–4339, 2013.

[27] T. Wongpiromsarn, U. Topcu, and R. Murray. Receding horizon temporal logic planning. *Automatic Control, IEEE Transactions on*, 57(11):2817–2830, 2012.