Barrier Functions: Bridging the Gap between Planning from Specifications and Safety-Critical Control

Petter Nilsson and Aaron D. Ames

***Abstract*—Real-life control systems are hierarchies of interacting layers; often consisting of a planning layer, a trajectory generation layer, and a trajectory-following layer. Independently designing the layers without taking the interactions between layers into account makes it difficult to obtain safety guarantees when executing a high-level plan. In this paper we combine ideas from safety-critical control and high-level policy synthesis to develop a principled connection between a high-level planner in a low-dimensional space, and a low-level safety-critical controller acting in the full state space. We introduce a new type of simulation relation and show that barrier functions can be used to abstract a high-dimensional system via the relation. As a result, we obtain provably correct execution of high-level policies by low-level optimization-based controllers. The results are demonstrated with a quadrotor surveillance example.**

I. INTRODUCTION

Safety is becoming an increasingly important factor in modern engineered systems. With evolving automation and the rise of autonomy, systems are expected to perform more complex tasks, and often operate in the vicinity of humans. This imposes a need for control design methodologies capable of guaranteeing correct and safe behaviors. A typical control architecture consists of multiple interacting layers. It is for example common to have a low-level layer with high-frequency feedback control, and other layers that generate plans and motions for the low-level controllers to track. A layered controller introduces additional complexity: how can the behaviors of the different layers and their interactions be restricted so as to guarantee safety? In this paper we propose a framework for addressing this question.

A popular method for guaranteeing safety in low-level control is *barrier functions*—Lyapunov-like certificates for set invariance [1]. However, by itself a barrier function only guarantees forward invariance and/or asymptotic stability of a set, which is insufficient to achieve complex behaviors. On the other hand, control synthesis utilizing temporal logic specifications has emerged as a popular way of designing plans for complex tasks [2], [3], [4]. Unfortunately, the synthesis problem is computationally difficult to solve, especially for high-dimensional highly dynamical models where typical grid-based abstractions methods are intractable due to the curse of dimensionality. Therefore, the synthesis problem is often solved on a low-fidelity model, and there are no guarantees that the plan can actually be implemented in a correct fashion.

The authors are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena CA 91125, U.S.A. Email addresses: {pettni, ames}@caltech.edu. This research was funded through the NASA JPL President's and Director's Fund Program.

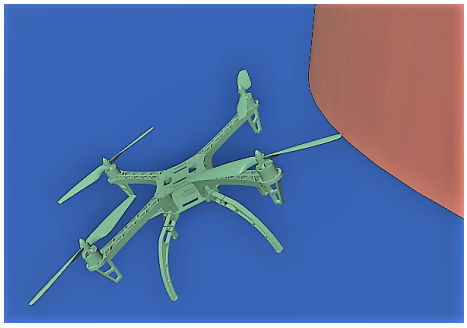

Fig. 1. Barrier function-based safety constraints prevent a quadrotor from hitting obstacles in a simulation with adverse wind conditions.

In this paper we combine barrier functions with ideas from temporal logic synthesis with the goal of leveraging strengths of both: provably correct high-fidelity implementation of temporal logic policies. We propose using barrier functions to control the higher-order dynamics and cancel out disturbances, thus leaving the high-level planner with an easier task. In addition, the barrier functions can be used as supervisors only, which leaves room for online trajectory generation that can improve performance while remaining in the safety envelope. Our method can be seen as an automated way of orchestrating enforcement of different barrier conditions over time, in a way that forces the system to satisfy a desired behavior. Simultaneous enforcement of multiple barrier conditions has been previously studied [5], [6], but scheduling barrier function enforcement over time so as to achieve a temporal task is to the best of our knowledge novel.

Comparing with existing methods, an alternative methodology for high-fidelity implementation of plans generated on a low-fidelity model was recently proposed by Herbert et al. [7], who treated the relation between the low-level controller and the planner as a two-player game. Also related is the work by Majumdar et al. on how *funnels* can be stitched together in a provably correct fashion [8]. We extend these works by considering planning with respect to temporal logic specifications, and leverage recent work on barrier functions that is suitable for decoupling stability/performance and safety [1], [9]. A different type of barrier certificate has previously been suggested as a tool in a proof system for verification [10].

In the following section we define (continuous) control systems and (finite) transition systems, and what it means for a system to satisfy an LTL specification. In Section III we propose a simulation relation that guarantees that an abstract

policy can be implemented on the concrete control system, and a way of constructing the abstraction by using barrier functions as certificates. We show an example in Section IV, and conclude the paper in Section V.

II. PRELIMINARIES

Below we define concepts related to temporal logics, control systems, and transition systems. These are all standard concepts from the literature, with one exception: specification satisfaction for transition systems is defined slightly differently to allow for non-deterministic labels and stuttering. This allows us to have fewer restrictions[1] on the barrier functions at the cost of slightly more complexity in the abstract system. We comment in Section II-E below on how to solve the abstract synthesis problem under this modification.

A. Notation

A relation $\mathcal{R}$ from $\mathcal{X}_1$ to $\mathcal{X}_2$ is a subset of $\mathcal{X}_1 \times \mathcal{X}_2$. For sets $X_1 \subset \mathcal{X}_1$ and $X_2 \subset \mathcal{X}_2$, we write $\mathcal{R}(X_1) = \{x_2 \in \mathcal{X}_2 \mid \exists x_1 \in X_1, x_1 \mathcal{R} x_2\}$ and $\mathcal{R}^{-1}(X_2) = \{x_1 \in \mathcal{X}_1 \mid \exists x_2 \in X_2, x_1 \mathcal{R} x_2\}$.

For a control- and disturbance-affine ODE $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = f(\mathbf{x}) + g_u(\mathbf{x})\mathbf{u} + g_d(\mathbf{x})\mathbf{d}$ and a function $h(t,x)$ we write the total time derivative of h under the ODE flow as

$$\mathcal{L}h(t,x,u,d) = \frac{\partial h}{\partial t}(t,x) + \mathcal{L}_f h(t,x) + \mathcal{L}_{g_u} h(t,x)u + \mathcal{L}_{g_d} h(t,x)d,$$

where $\mathcal{L}_f h(t,x) = \nabla_x h(t,x) \cdot f(x)$, and analogously for g_u and g_d. Note that depending on the relative degree of h, $\mathcal{L}h$ might not depend directly on all states in x and/or the inputs u and d.

B. Temporal Logics

Linear Temporal Logic (LTL) is a formalism for specifying temporal properties [11]. LTL has two types of operators: logical connectives and temporal modal operators. The logic connectives are those used in propositional logic: *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$) and *material implication* ($\implies$). The temporal modal operators are *always* ($\Box$), *eventually* ($\Diamond$), *next* ($\bigcirc$), and *until* ($\mathbf{U}$). In this paper we work with LTL over continuous time where the "next" ($\bigcirc$) operator lacks a meaningful interpretation. We therefore restrict attention to the "LTL without next" fragment of LTL, denoted LTL$_{\setminus\bigcirc}$. An LTL$_{\setminus\bigcirc}$ formula over a finite set of atomic propositions AP can be defined inductively as follows:

(i) $\texttt{True}$ is an LTL$_{\setminus\bigcirc}$ formula,
(ii) any atomic proposition $p \in AP$ is an LTL$_{\setminus\bigcirc}$ formula,
(iii) given LTL$_{\setminus\bigcirc}$ formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, and $\varphi \,\mathbf{U}\, \psi$ are also LTL$_{\setminus\bigcirc}$ formulas.

The remaining derived operators are defined as follows: (i) $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, (ii) $\varphi \implies \psi := \neg\varphi \vee \psi$, (iii) $\Diamond\varphi := \texttt{True}\ \mathbf{U}\ \varphi$, and (iv) $\Box\varphi := \neg\Diamond\neg\varphi$.

LTL formulas are interpreted over *ω-words*, which are infinite sequences in 2^{AP}. Given an ω-word $w = w(0)w(1)w(2)\ldots$ in 2^{AP} and an LTL formula φ, we write $(w,k) \models \varphi$ if and only if w satisfies φ at a position $k \geq 0$, which is defined inductively as follows:

(i) For an atomic proposition $p \in AP$, $(w,k) \models p$ iff $p \in w(k)$;
(ii) $(w,k) \models \neg\varphi$ iff $(w,k) \not\models \varphi$;
(iii) $(w,k) \models \varphi \vee \psi$ iff $(w,k) \models \varphi$ or $(w,k) \models \psi$;
(iv) $(w,k) \models \varphi \,\mathbf{U}\, \psi$ iff there exists $j \geq k$ such that $(w,j) \models \psi$ and $(w,i) \models \varphi$ for all $i \in [k,j)$.

An ω-word w satisfies φ, written as $w \models \varphi$, if and only if $(w,0) \models \varphi$.

We next define what it means for control systems and transition systems to satisfy an LTL property.

C. Control Systems and LTL Satisfaction

A *control system* Σ is a tuple $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, f, h_{\mathcal{X}})$ consisting of a state space $\mathcal{X}$, an initial set $\mathcal{X}_0 \subset \mathcal{X}$, an input set $\mathcal{U}$, a disturbance set $\mathcal{D}$, a parametrized vector field $f : \mathcal{X} \times \mathcal{U} \times \mathcal{D} \to T\mathcal{X}$, and an output map $h_{\mathcal{X}} : \mathcal{X} \to 2^{AP}$ [12]. In the following we assume that the sets $\mathcal{X}, \mathcal{X}_0, \mathcal{U}$ and $\mathcal{D}$ are compact. A *feedback controller* for Σ is a rule π_Σ that determines the input $\mathbf{u}(t)$ based on the current state $\mathbf{x}(t)$ and based on potential internal memory states[2]. A *trajectory* of Σ is an absolutely continuous function $\mathbf{x} : \mathbb{R}_+ \to \mathcal{X}$ such that

$$\mathbf{x}(0) \in \mathcal{X}_0,\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \text{ a.e.}, \qquad (1)$$

for some input and disturbance trajectories $\mathbf{u} : \mathbb{R}_+ \to \mathcal{U}$ and $\mathbf{d} : \mathbb{R}_+ \to \mathcal{D}$. In the following we assume that standard conditions are satisfied that guarantee uniqueness and maximality of an appropriate notion of solutions. If $\mathbf{u}(t)$ is obtained from a controller π_Σ, we say that the trajectory is *generated* by π_Σ.

Given a trajectory $\mathbf{x}(t)$ and an output map $h : \mathcal{X} \to 2^{AP}$, the *observation* of $\mathbf{x}$ is the trajectory $h_{\mathcal{X}} \circ \mathbf{x} : \mathbb{R}_+ \to 2^{AP}$. An ω-word $w = w(0)w(1)w(2)\ldots$ in 2^{AP} is said to be *consistent* with the output $h_{\mathcal{X}} \circ \mathbf{x}$ if and only if there exists a strictly increasing sequence $\{t_k\}_{k=0}^{\infty}$ in $[0,\infty)$ with $t_0 = 0$ and $t_k \longrightarrow \infty$ as $k \longrightarrow \infty$, such that $h_{\mathcal{X}} \circ \mathbf{x}(t) = w(k)$ for all $t \in [t_k, t_{k+1})$. Note that for any such sequence $\{t_k\}_{k=0}^{\infty}$, all ω-words consistent with the observation of $\mathbf{x}$ are stutter-equivalent [11]. We say that $\mathbf{x}$ *satisfies* φ, written $\mathbf{x} \models \varphi$, if there exists an ω-word w consistent with the observation of $\mathbf{x}$ such that $w \models \varphi$. If all trajectories of Σ generated by a controller π_Σ satisfy φ, we say that π_Σ *enforces* φ on Σ and write $(\Sigma, \pi_\Sigma) \models \varphi$.

D. Transition Systems and LTL Satisfaction

A *transition system* is a tuple $\mathcal{T} = (\mathbb{X}, \mathbb{X}_0, \mathbb{U}, \longrightarrow, h_{\mathbb{X}})$ where $\mathbb{X}$ is a set of states, $\mathbb{X}_0 \subset \mathbb{X}$ a set of initial states, $\mathbb{U}$ a set of inputs, $\longrightarrow \subset \mathbb{X} \times \mathbb{U} \times \mathbb{X}$ a transition relation, and $h_{\mathbb{X}} : \mathbb{X} \to 2^{2^{AP}}$ an output function [13]. A *policy* for $\mathcal{T}$ is a tuple $\pi_{\mathcal{T}} = (\pi_{\mathbb{U}}, \pi_{\mathbb{M}}, \mathbb{M}, m_0)$ where $\pi_{\mathbb{U}} : \mathbb{M} \times \mathbb{X} \to \mathbb{U}$ is a function generating inputs, $\pi_{\mathbb{M}} : \mathbb{M} \times \mathbb{X} \to \mathbb{M}$ is a memory

[1] In much literature, abstractions are required to be *proposition preserving* (e.g. [4]) which is inconvenient for the type of abstraction we consider here.

[2] The controllers we consider in this paper can be seen as hybrid feedback controllers, where the switch from one state-feedback controller $u_1(t,x)$ to the next controller $u_2(t,x)$ is triggered by guard conditions.

update function, $\mathbb{M}$ is a (finite) set of internal memory states, and $m_0 \in \mathbb{M}$ is the initial memory state.

As a shorthand notation, we write $\xi \xrightarrow{\mu} \xi'$ to indicate that $(\xi, \mu, \xi') \in \longrightarrow$. A *trajectory* of the transition system is a sequence $\zeta(0)\zeta(1)\zeta(2), \dots$ in $\mathbb{X}$ such that $\zeta(k) \in \mathbb{X}_0$ and such that $\zeta(k) \xrightarrow{\sigma(k)} \zeta(k+1)$ for some input trajectory $\sigma : \mathbb{N} \to \mathbb{U}$. A trajectory is *generated* by a policy $\pi_{\mathcal{T}}$ if $\sigma(k) = \pi_{\mathbb{U}}(m(k), \zeta(k))$ and the memory trajectory $m(k)$ is such that $m(0) = m_0$ and $m(k+1) = \pi_{\mathbb{M}}(m(k), \zeta(k))$.

For a trajectory $\zeta(0)\zeta(1)\zeta(2)\dots$ we can define the output $h_{\mathbb{X}} \circ \zeta : \mathbb{N} \to 2^{2^{AP}}$ that for each time instance k assigns a set $h_{\mathbb{X}} \circ \zeta(k) = \{l_i\}_i \in 2^{2^{AP}}$ of subsets $l_i \in 2^{AP}$ of atomic propositions. For a trajectory ζ we consider all finite words w of the form

$$w = w_0^0 w_0^1 \dots w_0^{c_0} w_1^0 w_1^1 \dots w_1^{c_1} w_2^0 w_2^1 \dots w_2^{c_2} \dots, \qquad (2)$$

where $w_k^{j_k} \in h_{\mathbb{X}} \circ \zeta(k)$ for $j_k = 0, \dots, c_k$ and $c_k < +\infty$. We say that ζ *satisfies* an LTL$_{\setminus\bigcirc}$ specification φ, written $\zeta \models \varphi$, if and only if $w \models \varphi$ for all words w on the form (2). If all $\pi_{\mathcal{T}}$-controlled trajectories of a transition system $\mathcal{T}$ satisfy φ, we say that $\pi_{\mathcal{T}}$ *enforces* φ on $\mathcal{T}$ and write $(\mathcal{T}, \pi_{\mathcal{T}}) \models \varphi$.

This definition of specification satisfaction differs in two ways from what is typically encountered in the literature: the output map $h_{\mathbb{X}}$ is non-deterministic in that it maps to $2^{2^{AP}}$ instead of to 2^{AP}, and there is a possibility of having multiple outputs for each time step (i.e. stuttering). Note that if $|h_{\mathbb{X}}(\xi)| = 1$ for all $\xi \in \mathbb{X}$, i.e., if there is no non-determinism, then all outputs of the form (2) are stutter-equivalent to the standard notion of output.

E. Solving a Synthesis Problem on a Transition System

The definition of specification satisfaction above is slightly more involved and requires modifications of discrete synthesis algorithms. One possibility is to decompose any state with multiple outputs (i.e. $|h_{\mathbb{X}}(\xi)| > 1$) into one state for each possible output, as illustrated in Figure 2. Transitions between the new states can then be added to capture stuttering, along with progress conditions that ensure that any stuttering lasts only a finite time. *Augmented* finite transitions systems [14] support such progress conditions, and the synthesis problem for augmented finite transition systems can be solved with tools such as ARCS [15]. An alternative to state splitting is to always assume the worst-case output [16].

While the complexity of the discrete synthesis problem is higher, in practice we expect most states to have a low number of possible outputs.

III. BARRIER FUNCTION-BASED PLANNING RELATIONS

Simulation relations have emerged as a tool to formalize the relationship between continuous systems and their abstractions [12], [13], [17], [18]. In most work on abstractions, the time scales of the continuous system and its abstraction are the same, in the sense that control input updates are done at the same sampling frequency. This is a shortcoming since low-level control loops typically need to run at a much higher frequency than high-level planners in order to accurately control nonlinear dynamics. Previous work is also largely concentrated on state space discretization. Although efficient abstractions techniques have been discovered for systems with special properties (e.g. monotonicity [19] or permutation invariance [20]), discretization of the entire state space is in general infeasible for high-dimensional systems.

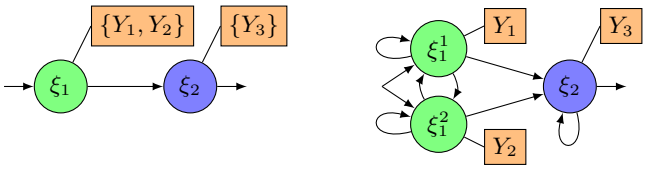

Fig. 2. Illustration of how a system with nondeterministic labels and stuttering (left) can be decomposed into a system with deterministic labels and no stuttering (right). In the illustration, $h(\xi_1) = \{Y_1, Y_2\}$, i.e. the output can be either Y_1 or Y_2 and the system can stutter between the two. The system to the right with $h(\xi_1^1) = Y_1$ and $h(\xi_1^2) = Y_2$ does not have nondeterministic labeling and produces the exact same outputs if the system is constrained to exit $\{\xi_1^1, \xi_1^2\}$ and then ξ_2^1 in finite time.

To address these shortcomings, and to pave the way for correct-by-construction planning for nonlinear high-dimensional systems, we introduce a simulation relation we call *alternating planning relation* that captures the relationship between a control system and an abstraction. This relation does not constrain the systems to operate at the same time scale, and relies on local feedback controllers to execute transitions between abstract states. This can be seen as a combination of elements of the time-agnostic over-approximations explored in [14], feedback-based roadmap techniques [21], and simulation relations with nondeterministic labeling [16]. We then propose barrier functions that serve as certificates for the conditions imposed by an alternating planning relation.

A. Alternating Planning Relations

We define a type of simulation relation from a control system $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, f, h_{\mathcal{X}})$ to a transition system $\mathcal{T} = (\mathbb{X}, \mathbb{X}_0, \mathbb{U}, \longrightarrow, h_{\mathbb{X}})$, where $h_{\mathcal{X}} : \mathcal{X} \to 2^{AP}$ and $h_{\mathbb{X}} : \mathbb{X} \to 2^{2^{AP}}$. The relation relies on the existence of feedback controllers that manage the internal dynamics to comply with the relation. As a consequence, by delegating control of internal dynamics to the local feedback controllers, the abstract state space $\mathbb{X}$ can be embedded in a low-dimensional subspace of $\mathcal{X}$ which allows constructing abstractions of moderate size even when $\mathcal{X}$ is high-dimensional.

Simulation relations often assume that systems execute concurrently, or, for the case of continuous-time systems, that sample times are fixed. Here we relax this condition which allows us to obtain sparser abstractions. As a result, there is no time equivalence between the two systems. Later in the paper we discuss how this shortcoming can be addressed by encoding side information in the abstract system $\mathcal{T}$. Notions of simulation relations that do not preserve time have previously been suggested for discrete systems. These include *weak* [22] (also known as *observable* [23]) simulations, that allow for silent actions to occur between regular events, and

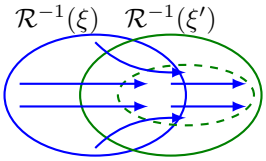

Fig. 3. Illustration of the requirements (3). For a discrete transition $\xi \to \xi'$, there must exist a feedback controller such that all points in $\mathcal{R}^{-1}(\xi)$ are steered to $\mathcal{R}^{-1}(\xi')$ in finite time, and the trajectories must remain in the union of these two sets.

stuttering [11] simulations. It has been shown using ideas similar to ours that hybrid systems that exhibit certain limit properties are weakly bisimilar to a finite system [24]. The notion of simulation relation we use is however closer to stuttering simulations in that there are no silent transitions in the abstraction.

Definition 1. Consider a control system $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, f, h_{\mathcal{X}})$ and a transition system $\mathcal{T} = (\mathbb{X}, \mathbb{X}_0, \mathbb{U}, \longrightarrow, h_{\mathbb{X}})$. A relation $\mathcal{R} \subset \mathcal{X} \times \mathbb{X}$ is an **alternating planning relation** from Σ to $\mathcal{T}$ if:

1) For all $x_0 \in \mathcal{X}_0$ there exists $\xi_0 \in \mathbb{X}_0$ such that $x_0 \mathcal{R} \xi_0$,
2) For all $x \mathcal{R} \xi$, $h_{\mathcal{X}}(x) \in h_{\mathbb{X}}(\xi)$,
3) For all $\xi \in \mathbb{X}$ and for all $\mu \in \mathbb{U}$ such that $\{\xi' : \xi \xrightarrow{\mu} \xi'\}$ is nonempty, there exists a feedback controller $u(t, x)$ such that the following holds: for all initial conditions $x_0 \in \mathcal{R}^{-1}(\xi)$ and disturbance signals $\mathbf{d}$, the resulting u-controlled trajectory $\mathbf{x}(t)$ with $\mathbf{x}(0) = x_0$ for some $0 \leq T < +\infty$ satisfies

$$\mathbf{x}(T) \in \bigcup_{\xi' : \xi \xrightarrow{\mu} \xi'} \mathcal{R}^{-1}(\xi'), \tag{3a}$$

$$\mathbf{x}([0, T)) \subset \mathcal{R}^{-1}(\xi) \cup \bigcup_{\xi' : \xi \xrightarrow{\mu} \xi'} \mathcal{R}^{-1}(\xi'). \tag{3b}$$

If there exists an alternating planning relation from Σ to $\mathcal{T}$ we write $\Sigma \preceq_{\text{plan}} \mathcal{T}$ and say that $\mathcal{T}$ **simulates** Σ.

Remark 1. The requirement in item 3 is illustrated in Fig. 3. The sets $\mathcal{R}^{-1}(\xi)$ are domains of the local feedback controllers, and (3) stipulates that the local feedback controllers should be able to steer Σ from an initial domain $\mathcal{R}^{-1}(\xi)$ to a final domain $\mathcal{R}^{-1}(\xi')$ whenever there exists a transition $\xi \xrightarrow{\mu} \xi'$ in $\mathcal{T}$. If the state-input pair (ξ, μ) has multiple successors, the requirement is that the domain $\mathcal{R}^{-1}(\xi')$ of at least one successor ξ' must be reached in finite time.

By construction, an alternating planning relation makes it possible to implement an abstract policy designed for $\mathcal{T}$ as a hybrid feedback controller for Σ in a way that preserves specification enforcement. The construction of the feedback controller is detailed in the proof of the following theorem.

Theorem 1. *Suppose that $\Sigma \preceq_{\text{plan}} \mathcal{T}$ and that $\pi_{\mathcal{T}} = (\pi_{\mathbb{U}}, \pi_{\mathbb{M}}, \mathbb{M}, m_0)$ is a policy for $\mathcal{T}$ such that $(\mathcal{T}, \pi_{\mathcal{T}}) \models \varphi$. Then there exists a controller π_Σ for Σ such that $(\Sigma, \pi_\Sigma) \models \varphi$.*

Proof. Take an initial state $x_0 \in \mathcal{X}_0$. By item 1 in Definition 1 there exists $\xi_0 \in \mathbb{X}_0$ such $x_0 \in \mathcal{R}^{-1}(\xi_0)$. Let $\pi_{\mathcal{T}} = (\pi_{\mathbb{U}}, \pi_{\mathbb{M}}, \mathbb{M}, m_0)$ be a policy that enforces φ on $\mathcal{T}$; we iteratively construct a trajectory ζ of $\mathcal{T}$ that is the discrete-time analogue of an evolving trajectory $\mathbf{x}$ of Σ with $\mathbf{x}(0) = x_0$. We let $\zeta(0) = \xi_0$ and $\sigma(0) = \pi_{\mathbb{U}}(\xi_0, m_0)$.

Assume by induction that at continuous time instant t_k the discrete state is $\zeta(k)$, that the controller memory state is $m(k)$, and that $\mathbf{x}(t_k) \in \mathcal{R}^{-1}(\zeta(k))$. We select the discrete input $\sigma(k) = \pi_{\mathbb{U}}(m(k), \zeta(k))$ and update the memory state according to $m(k+1) = \pi_{\mathbb{M}}(m(k), \zeta(k))$. By item 3 in Definition 1 we can find a feedback controller such that the set $\bigcup_{\xi' : \zeta(k) \xrightarrow{\sigma(k)} \xi'} \mathcal{R}^{-1}(\xi')$ is reached in finite time, and such that $\mathbf{x}(t)$ remains in $\mathcal{R}^{-1}(\zeta(k))$ until that set is reached. Assume that the set is reached at a time t_{k+1} and select $\zeta(k+1)$ such that $\mathbf{x}(t_{k+1}) \in \mathcal{R}^{-1}(\zeta(k+1))$. This shows that we can inductively construct a trajectory ζ with the property that

$$\mathbf{x}(t) \mathcal{R} \zeta(k), \quad \forall t \in [t_k, t_{k+1}). \tag{4}$$

Consider now the output $h_{\mathcal{X}} \circ \mathbf{x}$ on the interval $[t_k, t_{k+1})$. The interval $[t_k, t_{k+1})$ can be partitioned into intervals $[t_k^j, t_k^{j+1})$ such that

$$h_{\mathcal{X}}(\mathbf{x}(t)) = w_k^j \quad \forall t \in [t_k^j, t_k^{j+1}). \tag{5}$$

In other words, the word w_k^j is consistent with the output $h_{\mathcal{X}} \circ \mathbf{x}$. Furthermore, the word is on the form (2), and, by item 2 in Definition 1, $w_k^j \in h_{\mathbb{X}}(\zeta(k))$. Therefore the word w_k^j is also a word of $\mathcal{T}$ generated by $\pi_{\mathcal{T}}$. Since the policy $\pi_{\mathcal{T}}$ enforces φ on $\mathcal{T}$, it follows that the feedback controllers as scheduled above enforce φ on Σ. □

Given a control system Σ, constructing an abstraction $\mathcal{T}$ such that $\Sigma \preceq_{\text{plan}} \mathcal{T}$ requires designing local feedback controllers. Barrier functions [1] are a convenient tool to reason about the dynamics of set membership constraints using Lyapunov-like certificates. In the following we give a set of barrier function conditions that imply the conditions in Definition 1.

B. Barriers Functions for Invariance and Reachability

Consider a system $\Sigma = (\mathcal{X} \times \mathcal{V}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, f, h_{\mathcal{X}})$ where $h_{\mathcal{X}} : \mathcal{X} \to 2^{AP}$ and f is an affine control system of the form

$$\Sigma : \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} = f(\mathbf{x}, \mathbf{v}) + g_u(\mathbf{x}, \mathbf{v})\mathbf{u} + g_d(\mathbf{x}, \mathbf{v})\mathbf{d}. \tag{6}$$

The state space of Σ is $\mathcal{X} \times \mathcal{V}$, where $\mathcal{X}$ is the *planning space* and $\mathcal{V}$ is the *internal dynamics* space that governs the motion in planning space. For instance, motion planning for mobile robots is usually done on the robot pose space while assuming that the generated plan can be implemented by low-level controllers that control the velocity states. In addition, specifications are often expressed in terms of pose only, and, if specifications regarding velocity are present, they are typically of invariance type (e.g., a maximal allowed velocity), which is best dealt with at the low level anyway. This motivates the decomposition into a planning space and an internal dynamics space, and is the reason why we have

defined the output function $h_{\mathcal{X}}$ to map from $\mathcal{X}$ instead of from $\mathcal{X} \times \mathcal{V}$.

We now show how an alternating planning relation can be constructed by using barrier functions as certificates for (3). For scalability purposes we want to restrict planning to the state space $\mathcal{X}$; to implement a plan in a provably safe manner the auxiliary states $\mathbf{v}$ must be controlled by local feedback controllers. As it turns out, barrier functions for systems with relative degree are ideally suited for this task since they mimic the cascaded structure of many robotic systems.

To start with, we assume that for a collection of compact sets $\{X_i\}_{i\in I}$ in $\mathcal{X}$, there are local invariance-enforcing barrier functions h_i^{inv} such that

$$h_i^{\text{inv}}(x,v) \geq 0 \implies x \in X_i, \tag{7a}$$
$$\kappa_i^{\text{inv}}(h_i^{\text{inv}}(x,v)) + \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \mathcal{L}h_i^{\text{inv}}(x,v,u,d) \geq 0, \tag{7b}$$

where κ_i^{inv} is a class-$\mathcal{K}$ function. We build on the definition of a (zeroing) control barrier function [1] wherein a set $C = \{x \in X : h(x) \geq 0)$ is safe (forward invariant) if $\kappa(h(\mathbf{x})) + \frac{\mathrm{d}}{\mathrm{d}t}h(\mathbf{x}) \geq 0$ for a class-$\mathcal{K}$ function κ. In this case, our safe set is a set in $x - v$ space that by (7a) projects inside of X_i. This allows us to enforce invariance of X_i via the internal dynamics provided that control inputs satisfying (7b) can be synthesized.

Lemma 1. *A barrier function on the form (7) renders a subset of X_i controlled invariant.*

Proof. We show that the set $C = \{(x,v) : h_i^{\text{inv}}(x,v) \geq 0\}$ is controlled invariant, which is a subset of X_i via (7a). For a smooth h_i^{inv} controlled invariance follows from Nagumo's theorem [25]: from (7b) it is always possible to select $u(x,v)$ such that for all $d \in \mathcal{D}$,

$$\frac{\mathrm{d}}{\mathrm{d}t}h_i^{\text{inv}}(x,v,u(x,v),d) \geq -\kappa_i^{\text{inv}}(h_i^{\text{inv}}(x,v)). \tag{8}$$

From this inequality it follows that if $h_i^{\text{inv}}(x,v) = 0$, then $\frac{\mathrm{d}}{\mathrm{d}t}h_i^{\text{inv}}(x,v,u,d) \geq 0$, which implies forward invariance of C. □

Next, we seek to connect the sets $\{X_i\}_{i\in I}$ with barrier functions that steer the system from one set to the next. A function $h_{ij}^{\text{rch}}(t,x,v)$ is a certificate that the system can be steered from X_i to X_j in time T_{ij} if for all x, v and all $t \in [0, T_{ij}]$:

$$h_i^{\text{inv}}(x,v) \geq 0 \implies h_{ij}^{\text{rch}}(0,x,v) \geq 0, \tag{9a}$$
$$h_{ij}^{\text{rch}}(T_{ij},x,v) \geq 0 \implies h_j^{\text{inv}}(x,v) \geq 0, \tag{9b}$$
$$\kappa_{i,j}^{\text{rch}}(h_{ij}^{\text{rch}}(t,x,v)) + \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \mathcal{L}h_{ij}^{\text{rch}}(t,x,v,u,d) \geq 0. \tag{9c}$$

Lemma 2. *Barriers on the form (9) render the set $C_t = \{(x,v) : \exists t \in [0,T_{ij}], h_{ij}^{\text{rch}}(t,x,v) \geq 0\}$ controlled invariant, and render the set $\{(x,v) : h_{ij}^{\text{rch}}(T_{ij},x,v) \geq 0\}$ reachable (without leaving C_t) from initial conditions in C_t.*

Proof. By the same argument as in Lemma 1 it follows that the set $C = \{(t,x,v) : h_{ij}^{\text{rch}}(t,x,v) \geq 0\}$ is controlled invariant. This fact implies both claims in the lemma: C_t is the projection of C and $\{(x,v) : h_{ij}^{\text{rch}}(T_{ij},x,v) \geq 0\}$ is a slice of C at time $t = T_{ij}$. □

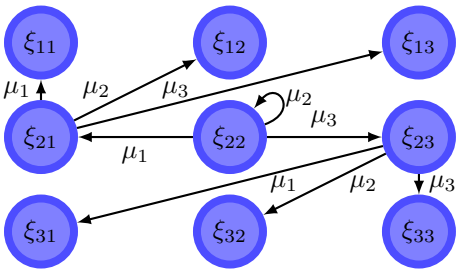

Fig. 4. Illustration of abstraction $\mathcal{T}$ when $I = \{1,2,3\}$. Only transitions going out from the middle states ξ_{21}, ξ_{22} and ξ_{23} are shown. If the abstract state is ξ_{21} (i.e., the underlying system is in the process of steering from X_2 to X_1), selecting abstract input 1 implies staying in X_1, selecting abstract input 2 implies steering back to X_2, and selecting abstract input 3 implies steering to X_3.

Remark 2. In theory, the same barrier certificate could be used for enforcing both invariance and reachability, i.e. $h_{ij}^{\text{rch}}(T,x,v) = h_j^{\text{inv}}(x,v)$. However, additional degrees of freedom can make (9) easier to satisfy and improve performance. In addition, these ideas generalize to more than two barrier functions: different reachability barrier functions could be used to connect sets along different paths.

C. Abstraction Construction

From barrier functions like these we construct an abstraction $\mathcal{T} = (\mathbb{X}, \mathbb{X}_0, \mathbb{U}, \longrightarrow, h_{\mathbb{X}})$. The sets $\{X_i\}_{i\in I}$ should be thought of as margins around sample points used in a planning algorithm like RRT [26] that constructs sparse graph structures. While it would be natural to think of the sets $\{X_i\}$ as abstract states, this would disregard time spent transitioning from one set X_i to another set X_j. Since events during transitions are equally important to capture, we define abstract states ξ_{ij} that correspond to being in the process of transitioning from X_i to X_j. With this construction, an abstract state ξ_{ii} corresponds to (a subset of) X_i being kept invariant.

Abstract states and inputs: Abstract states are of the form ξ_{ij} for $i,j \in I$, thus the abstract state space is $\mathbb{X} = \{\xi_{ij}\}_{(i,j)\in I\times I}$. The input set is $\mathbb{U} = \{\mu_i\}_{i\in I}$.

Transition relation: We encode two types of transitions as illustrated in Fig. 4. Firstly, for $i \neq j$ we encode a transition $\xi_{ki} \xrightarrow{\mu_j} \xi_{ij}$ if and only if there exists a barrier certificate h_{ij}^{rch} that satisfies the properties in (9). Furthermore, we encode a transition $\xi_{ki} \xrightarrow{\mu_i} \xi_{ii}$ for each transition of the first type.

Remark 3. The discrete state space as defined above is of size $|I|^2$ and the input space is of size $|I|$. However, a smaller representation can often be obtained since all sets are typically not connected. Non-connected parts of the transition graph can be purged, and an optimally sized representation has a number of inputs equal to the maximal number of outward connections from any set X_i.

Planning relation and output map: We define a relation $\mathcal{R} \subset \mathcal{X} \times \mathbb{X}$ such that

$$\begin{aligned}(x,v)\mathcal{R}\xi_{ii} &\iff h_i^{\text{inv}}(x,v) \geq 0,\\ (x,v)\mathcal{R}\xi_{ij} &\iff \exists t \in [0,T_{ij}] \text{ s.t. } h_{ij}^{\text{rch}}(t,x,v) \geq 0.\end{aligned} \tag{10}$$

Based on the relation $\mathcal{R}$ we finally define the abstract input set as $\mathbb{X}_0 = \mathcal{R}(\mathcal{X}_0)$ and the output map $h_{\mathbb{X}}$ as

$$h_{\mathbb{X}}(\xi_{ij}) = \{h_{\mathcal{X}}(x) : \exists v,\ (x,v)\mathcal{R}\xi_{ij}\}. \tag{11}$$

Theorem 2. *Assume that $\mathcal{X}_0 \subset \bigcup_{\xi\in\mathbb{X}} \mathcal{R}^{-1}(\xi)$. Then $\mathcal{R}$ is an alternating planning relation from Σ to $\mathcal{T}$ and thus $\Sigma \preceq_{\text{plan}} \mathcal{T}$.*

Proof. Condition 1 of Definition 1 holds by the assumption in the theorem and condition 2 holds by definition due to (11). What remains is to show condition 3. To this end, take $\xi_{ij} \in \mathbb{X}$ and $\mu_k \in \mathbb{U}$ such that $\xi_{ij} \xrightarrow{\mu_k} \xi_{jk}$ (i.e. μ_k is defined at ξ_{ij}). We need to show that any trajectory originating in $\mathcal{R}^{-1}(\xi_{ij})$ can be controlled in finite time to $\mathcal{R}^{-1}(\xi_{jk})$ without leaving $\mathcal{R}^{-1}(\xi_{ij})$.

Suppose that $j \neq k$. We know that there exist barrier functions h_{ij}^{rch}, h_j^{inv} and h_{jk}^{rch} that satisfy the properties (7) and (9). Thus, by Lemma 2 the set $C = \{(x,v) : h_{ij}^{\text{rch}}(T_{ij},x,v) \geq 0\}$ can be reached in finite time from $\mathcal{R}^{-1}(\xi_{ij})$. Furthermore, by (9b) and (9a),

$$\begin{aligned}(x,v) \in C \implies & h_j^{\text{inv}}(x,v) \geq 0\\ \implies & h_{jk}^{\text{rch}}(0,x,v) \geq 0 \implies (x,v) \in \mathcal{R}^{-1}(\xi_{jk}).\end{aligned}$$

Thus (3) is satisfied. The case $j = k$ follows from the first implication since $\mathcal{R}^{-1}(\xi_{jj}) = \{(x,v) : h_j^{\text{inv}}(x,v) \geq 0\}$. □

D. Discussion

The results presented above give conditions in the form of existence of barrier functions. In order to leverage the results for a given problem, functions satisfying the conditions must somehow be obtained. Finding a barrier function that enforces invariance is in general a challenging task—especially for systems with constrained input sets. There are however practical methods for finding barrier functions for systems with relative degree [9], and they can be computed using sums-of-squares programming [27]. To avoid having to construct unique barrier functions for each pair of sets X_i, X_j, a reasonable approach—that we use in the example in Section IV—is to design a single barrier function $h(x,v;x_0,\epsilon)$ that enforces invariance of a set $\{x : \|x - x_0\| \leq \epsilon\}$. The time-dependent barrier functions for reachability can then be defined as $h_{ij}^{\text{rch}}(t,x,v) = h(x,v;x_0(t),\epsilon(t))$, where $x_0(t)$ describes a curve connecting x_i to x_j. If $x_0(t)$ varies slowly enough, and if h satisfies the barrier condition robustly, then also $h_{ij}^{\text{rch}}(t,x,v)$ satisfies the barrier condition. This approach works if the dynamics are translational-invariant in x; other types of symmetries can potentially be exploited in the same manner.

This approach to provably correct high-level planning fits well together with incremental sample-based algorithms such as RRT; for such methods the abstraction can be incrementally refined until a policy is found. When a policy has been constructed it must be implemented as a controller in continuous time. The proof of Theorem 1 details how the abstract policy should be refined, which results in a controller that schedules enforcement of barrier function constraints. Barrier function constraints by themselves do not however guarantee stability. It is therefore practical to connect the barrier to a nominal controller that induces stability. This can be done by letting the barrier supervise a nominal controller, or by designing a control Lyapunov function and enforcing the resulting constraint together with the barrier condition [1]. Both methods can be implemented via online quadratic programming.

As mentioned above, there is no time equivalence between the concrete system and its abstraction in this method. This is not an issue for temporal logic planning in $\text{LTL}_{\setminus\bigcirc}$ since this fragment does not contain time constraints. However, in practice it is desirable to synthesize controllers that make progress as quickly as possible towards specification subgoals, and our method can potentially be used also with logics such as Signal Temporal Logic that do allow for expressing specifications over time. The timing issue can be mitigated by amending the abstract transitions with "weights" that represent the time needed for traversal, and utilize discrete synthesis techniques that compute the "time-to-reach" via value function iteration [28].

The type of abstraction considered in this paper is sound, in the sense that if $\Sigma \preceq_{\text{plan}} \mathcal{T}$, then a policy for $\mathcal{T}$ can be implemented on Σ with preserved correctness guarantees. As opposed to work on bisimulations [2] and approximate bisimulations [12], there is however no guarantee of finding a policy whenever one exists. Our method should be seen as a practical alternative that searches for solutions in a sparse space, and as a way to achieve a principled separation between high-level planning and low-level feedback control.

IV. EXAMPLE: BARRIER-CERTIFIED QUADROTOR PLANNING

We demonstrate the method on a planning problem for a quadrotor. The standard model for quadrotors is 12-dimensional and nonlinear—beyond the capabilities of typical abstraction methods. To circumvent the scalability issue, we design barrier functions that allow us to conduct mission planning in three-dimensional position space.

Dynamical equations for a quadrotor are obtained from force-balance equations in a rotating reference frame (e.g. [29], [30]). Let the 12-dimensional state be $\mathbf{x} = (\mathbf{r}, \mathbf{v}, \xi, \omega)$, where $\mathbf{r}$ and $\mathbf{v}$ are position and velocity in $\mathbb{R}^3$, $\xi = (\phi,\theta,\psi) \in SO(3)$ are roll, pitch and yaw angles, and $\omega \in \mathbb{R}^3$ are angular velocities in the quadrotor body frame. Then

$$\begin{aligned}&m\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r} = -mge_3 + f_z R(\xi)e_3 + d,\\ &\frac{\mathrm{d}}{\mathrm{d}t}\xi = T(\xi)\omega, \quad J\frac{\mathrm{d}}{\mathrm{d}t}\omega = \tau - (\omega \times (J\omega)).\end{aligned} \tag{12}$$

Here $e_3 = [0,0,1]^T$ and $R(\xi) = R_z(\psi)R_y(\theta)R_x(\phi)$ is the x-y-z rotation matrix from a body-fixed frame to the world frame, and $T(\xi)$ the corresponding mapping between angular velocities.

TABLE I
PARAMETER VALUES FOR THE QUADROTOR EXAMPLE.

$m = 5$ kg	$\epsilon = 2,\ \beta = 0.05$
$g = 9.81$ m/s^2	$a_1 = 0.25, a_2 = 2, a_3 = -2$
$J = 0.01 \times I_3$ kgm^2	$\gamma_1 = 1.5, \gamma_2 = 4.5$

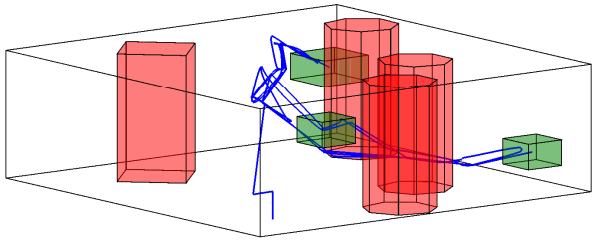

Fig. 5. Illustration of quadrotor environment with surveillance targets (green) and obstacles (red). The blue trajectory satisfies the specification φ by repeatedly visiting all targets without hitting obstacles.

We consider sets in the state space defined as $X_i = \{r : \|r - r_i\| \leq \epsilon\}$ for a finite set of waypoints r_i, where $\|\cdot\|$ is a weighted Euclidean norm. Inspired by the barrier function for obstacle avoidance proposed in [31], we utilize time-dependent barrier functions on the following form for $0 < \beta < 1$ and $a_1 < 1$:

$$h_0(t, r, \xi) = \epsilon^2(1-\beta) - \|r - r_0(t)\|^2 - \frac{\epsilon^2\beta}{\pi/2} a_1 \arctan\left(a_2(r - r_0(t))^T R(\xi) e_3 + a_3\right), \tag{13}$$

where $r_0(0) = r_i$ and $r_0(T_{ij}) = r_j$ for some r_i, r_j, T_{ij}.

It can be shown easily that if $h_0(t, r, \xi) \geq 0$, then $\|r - r_0(t)\| \leq \epsilon$, so $h_i^{inv}(r, \xi) = h_0(0, r, \xi)$ satisfies (7a) and $h_{ij}^{rch}(t, r, \xi) = h_0(t, r, \xi)$ satisfies (9a)-(9b). That is, we use functions on the form (13) both for invariance and reachability. In addition, h_0 has relative degree two with respect to (12), so we can define a new relative degree one function h_1 as

$$h_1(t, r, v, \xi, \omega) = \gamma_1 h_0(t, r, v) + \mathcal{L}h_0(t, r, v, \xi, \omega). \tag{14}$$

If the input set is unbounded, h_1 satisfies conditions (7b) and (9c) everywhere except in certain degenerate configurations[3]. In the following we set $\epsilon = 2$ and tune the barrier function so that it rejects wind disturbances gracefully. Parameter values for both the dynamics and the barrier function are listed in Table I.

We consider a surveillance scenario where the specification is to repeatedly visit target areas T_i, $i = 1, 2, 3$ while avoiding obstacles areas:

$$\varphi = \square(r \notin \text{Obs}) \wedge \bigwedge_{i=1}^{3} \square\lozenge(r \in T_i). \tag{15}$$

[3]From calculations similar to [31] it follows that the barrier function condition is only violated in certain measure-zero geometrical configurations. We don't expect such degenerate sets to be stable, or even reachable from "normal" initial conditions when the barrier condition is imposed, but this remains to be proven.

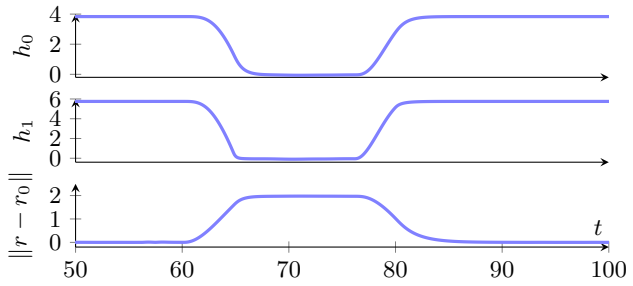

Fig. 6. Plot of barrier function values h_0 and h_1 over time, and the distance $\|r - r_0\|$ from the nominal position. As can be seen, the barrier functions remain positive which ensures that $\|r - r_0\| \leq 2$, even during the wind gust that affects the quadrotor from $t = 60$ to $t = 80$.

The environment contains four obstacles and three target regions, as shown in Fig. 5. We opt for a sample-based planning algorithm that adds waypoints $r_i \in \mathbb{R}^3$ and connects them via time-dependent barrier functions on the form (13)-(14) for r_0 being interpolated between the waypoints, i.e. $r_0(t) = r_i(1 - t_p(t)) + t_p(t) r_j$ where $t_p(t) \in [0, 1]$ is a phase variable that parametrizes time. These barriers allow us to construct an abstraction $\mathcal{T}$ with the property that $\Sigma \preceq_{\text{plan}} \mathcal{T}$, by over-approximating the sets (10) with polyhedral sets and checking for intersection with regions of interest (targets and obstacles). Thus, by adding additional waypoints until we can find a policy $\pi_{\mathcal{T}}$ such that $(\mathcal{T}, \pi_{\mathcal{T}}) \models \varphi$, we can refine the abstract policy to a controller π_Σ such that $(\Sigma, \pi_\Sigma) \models \varphi$. To solve the abstract synthesis problem we use the tool ARCS [15] which supports augmented finite transition systems as discussed in Section II-E. We opt for an implementation that separates performance and safety: we use a geometrical controller for tracking [32] that is supervised by barrier functions. The control inputs are computed via:

$$\begin{aligned} \min_{f_z, \tau} \ & \lambda_f (f_z - f_z^*)^2 + \|\tau - \tau^*\|^2, \\ \text{s.t.} \ & \gamma_2 h_1(t, r, v, \xi, w) + \min_{d \in \mathcal{D}} \mathcal{L}h_1(t, r, v, \xi, w, f_z, \tau, d) \geq 0, \end{aligned}$$

where f_z^* and τ^* are the nominal control inputs for tracking. For a bounded disturbance set, this ensures forward invariance of the set where h_0 and h_1 are positive. In our experiment we used $\mathcal{D} = \{(d_1, d_2, d_3) : |d_i| \leq 30\}$ for an assumed maximal disturbance of 30 N in any direction.

We tested the controller in a Simulink Simscape Multibody simulation. The blue trace in Fig. 5 highlights a 300s long execution of the controller through the environment, under a time-varying wind disturbance from the east with a maximal amplitude of 30 N. Barrier function values for a portion of the flight are shown in Fig. 6; as can be seen the barrier functions remain positive despite wind disturbance. The nominal controller was adequate during periods without wind disturbance, but for significant wind gusts the tracking deviates significantly from the nominal trajectory which implies a risk of hitting obstacles. Fig. 7 illustrates how nominal inputs are used during normal operation, but in adverse wind conditions the barrier constraint overrides the

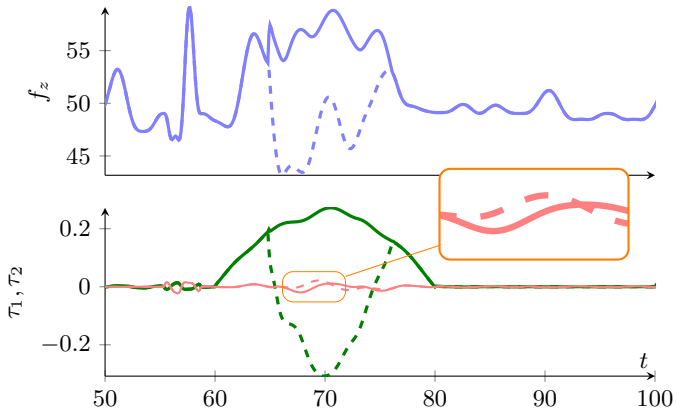

Fig. 7. Plot of nominal control inputs (dashed lines) and actual control inputs after supervision (solid lines). When the wind is strong the barrier constraint overrides nominal control action to ensure safety. However, when the wind disappears control is handed back to the nominal controller.

desired input. Fig. 1 shows the simulated quadrotor when the barrier constraint is active.

V. CONCLUSION

In this work we sought to connect high-level planning and low-level safety-critical control by leveraging barrier functions as the bridge. We gave an alternating simulation relation that separates responsibility between a high-level planner and low-level feedback controllers, and conditions based on barrier functions for satisfying the alternating simulation relation. We showcased the method on a quadrotor planning problem where our method allowed us to conduct planning in three-dimensional space but obtain guarantees with respect to the 12-dimensional nonlinear model.

In the future we will investigate how barrier functions can be constructed in a minimally intrusive way while still satisfying the conditions outlined in this paper, with the goal of achieving seamless switching from one barrier function constraint to the next. An interesting challenge for future work is to incorporate reactive elements, by for instance considering environments that change with time. Whether such changes should be treated in the planning layer or at a lower level depends on the time characteristics and predictability of the changes. We are also working towards testing these ideas on real hardware.

Acknowledgment: The authors would like to thank Paulo Tabuada for helpful discussions during the preparation of this work.

REFERENCES

[1] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[2] P. Tabuada and G. J. Pappas, "Linear Time Logic Control of Discrete-Time Linear Systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

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

[4] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[5] X. Xu, "Constrained control of input–output linearizable systems using control sharing barrier functions," *Automatica*, vol. 87, pp. 195–201, 2018.

[6] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *Proc. IEEE CDC*, 2016, pp. 2659–2664.

[7] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: A modular framework for fast and guaranteed safe motion planning," in *Proc. IEEE CDC*. IEEE, 2017, pp. 1517–1522.

[8] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[9] Q. Nguyen and K. Sreenath, "Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints," *Proc. ACC*, no. 3, pp. 322–328, 2016.

[10] R. Dimitrova and R. Majumdar, "Deductive control synthesis for alternating-time logics," in *Proc. EMSOFT*, 2014.

[11] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[12] G. Pola and P. Tabuada, "Symbolic Models for Nonlinear Control Systems: Alternating Approximate Bisimulations," *SIAM Journal on Control and Optimization*, vol. 48, no. 2, pp. 719–733, 2009.

[13] P. Tabuada, *Verification and Control of Hybrid Systems*. Springer, 2009.

[14] P. Nilsson, N. Ozay, and J. Liu, "Augmented finite transition systems as abstractions for control synthesis," *Discrete Event Dynamic Systems*, vol. 27, no. 2, pp. 301–340, 2017.

[15] O. B. Lindvall, P. Nilsson, and N. Ozay, "Nonuniform abstractions, refinement and controller synthesis with novel bdd encodings," in *Proc. IFAC ADHS*, 2018, pp. 19–24.

[16] S. Haesaert, P. Nilsson, C.-I. Vasile, R. Thakker, A. akbar Agha-Mohammadi, A. D. Ames, and R. M. Murray, "Temporal logic control of pomdps via label-based stochastic simulation relations," in *Proc. IFAC ADHS*, 2018, pp. 271–276.

[17] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 116–126, 2010.

[18] M. Zamani, A. Abate, and A. Girard, "Symbolic models for stochastic switched systems: A discretization and a discretization-free approach," *Automatica*, vol. 55, pp. 183–196, 2015.

[19] E. S. Kim, M. Arcak, and S. A. Seshia, "Symbolic control design for monotone systems with directed specifications," *Automatica*, vol. 83, pp. 10–19, 2017.

[20] P. Nilsson and N. Ozay, "Control synthesis for high-dimensional systems with counting constraints," *arXiv:1706.07863v1 [cs.SY]*, 2017.

[21] A.-a. Agha-mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[22] R. Milner, *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.

[23] C. Stirling, *Modal and temporal logics for processes*. Springer, 1996, pp. 149–237.

[24] H. G. Tanner, J. Fu, C. Rawal, J. L. Piovesan, and C. T. Abdallah, "Finite abstractions for hybrid systems with stable continuous dynamics," *Discrete Event Dynamic Systems*, vol. 22, no. 1, pp. 83–99, 2012.

[25] W. Walter, *Ordinary Differential Equations*. Springer, 1998.

[26] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[27] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness Guarantees for the Composition of Lane Keeping and Adaptive Cruise Control," *IEEE Trans. Autom Sci. Eng.*, pp. 1–14, 2017.

[28] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient Reactive Controller Synthesis for a Fragment of Linear Temporal Logic," in *Proc. IEEE ICRA*, 2013, pp. 5033–5040.

[29] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE ICRA*, 2011, pp. 2520–2525.

[30] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," *Proc. IEEE ICRA*, pp. 6567–6572, 2014.

[31] G. Wu and K. Sreenath, "Safety-Critical Control of a 3D Quadrotor With Range-Limited Sensing," in *Proc. ASME CDSC*, 2016, p. V001T05A006.

[32] T. Lee, M. Leok, and N. H. McClamroch, "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)," *arXiv:1003.2005 [math.OC]*, 2011.