Automatica 45 (2009) 343–352

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Temporal logic motion planning for dynamic robots☆

Georgios E. Fainekos [a,*], Antoine Girard [b], Hadas Kress-Gazit [a], George J. Pappas [a]

[a] 3330 Walnut Street, GRASP Laboratory, University of Pennsylvania, PA 19104, United States

[b] Laboratoire Jean Kuntzmann, Université Joseph Fourier, France

ARTICLE INFO

Article history:
Received 28 December 2006
Received in revised form
7 May 2008
Accepted 15 August 2008
Available online 17 December 2008

Keywords:
Motion planning
Temporal logic
Robustness
Hybrid systems
Hierarchical control

ABSTRACT

In this paper, we address the temporal logic motion planning problem for mobile robots that are modeled by second order dynamics. Temporal logic specifications can capture the usual control specifications such as reachability and invariance as well as more complex specifications like sequencing and obstacle avoidance. Our approach consists of three basic steps. First, we design a control law that enables the dynamic model to track a simpler kinematic model with a globally bounded error. Second, we built a robust temporal logic specification that takes into account the tracking errors of the first step. Finally, we solve the new robust temporal logic path planning problem for the kinematic model using automata theory and simple local vector fields. The resulting continuous time trajectory is provably guaranteed to satisfy the initial user specification.

1. Introduction

One of the main challenges in robotics is the development of mathematical frameworks that formally and verifiably integrate high level planning with continuous control primitives. Traditionally, the path planning problem for mobile robots has considered reachability specifications of the form "move from the Initial position I to the Goal position G while staying within region R". The solutions to this well-studied problem span a wide variety of methods, from continuous (such as potential or navigation functions (Choset et al., 2005, Section 4)) to discrete (such as Canny's algorithm, Voronoi diagrams, cell decompositions and probabilistic roadmaps (Choset et al., 2005; LaValle, 2006)).

Whereas these methods solve the basic path planning problem, they do not address high level planning issues that arise when one considers a number of goals or a particular ordering of them. In order to manage such constraints, one should employ one of the existing high level planning methods (LaValle, 2006). Even though the aforementioned methods can handle partial ordering of goals, they cannot deal with temporally extended goals. For such specifications, planning techniques (De Giacomo & Vardi, 1999) that are based on model checking (Clarke, Grumberg, & Peled, 1999) seem to be a better choice. Using temporally extended goals, one would sacrifice some of the efficiency of the standard planning methods for expressiveness in the specifications. Temporal logics such as Linear Temporal Logic (LTL) (Pnueli, 1977) and its continuous time version propositional temporal logic over the reals (RTL) (Reynolds, 2001) have the expressive power to describe a conditional sequencing of goals under a well-defined formal framework.

Such a formal framework can provide us with the tools for automated controller synthesis and code generation. Beyond the provably correct synthesis of hybrid controllers for path planning from high level specifications, temporal logics have one more potential advantage when compared to other formalisms, e.g., regular languages (Koutsoukos, Antsaklis, Stiver, & Lemmon, 2000). That is to say, temporal logics were designed to bear a resemblance to natural language. Along the same lines, one can develop computational interfaces between natural language and temporal logics (Kress-Gazit, Fainekos, & Pappas, 2007). This fact alone makes temporal logics a suitable medium for our daily discourse with future autonomous agents.

In our previous work (Fainekos, Kress-Gazit, & Pappas, 2005), we have combined such planning frameworks with local controllers defined over convex cells (Belta, Isler, & Pappas, 2005; Conner, Rizzi, & Choset, 2003) in order to perform temporal logic

☆ This research is partially supported by NSF EHS 0311123, NSF ITR 0121431 and ARO MURI DAAD 19-02-01-0383. The material in this paper was presented at the 44th IEEE Conference on Decision and Control, Seville, Spain, 12–15 December 2005 and Hybrid Systems: Computation and Control, Pisa, Italy, 3–5 April 2007. This paper was recommended for publication in revised form by Associate Editor Bart De Schutter under the direction of Editor Ian R. Petersen.

* Corresponding address: NEC Laboratories America, 4 Independence Way, Suite 200, 08540 Princeton, NJ, United States. Tel.: +1 609 951 2657; fax: +1 609 951 2481.

E-mail addresses: fainekos@grasp.upenn.edu, fainekos@seas.upenn.edu (G.E. Fainekos), antoine.girard@imag.fr (A. Girard), hadaskg@grasp.upenn.edu (H. Kress-Gazit), pappasg@grasp.upenn.edu (G.J. Pappas).

0005-1098/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2008.08.008

motion planning for a fully actuated kinematics model of a robot. In a kinematics model, the control inputs to the system are actually the desired velocities. However, when the velocity of the mobile robot is high enough, a kinematics model is not enough any more, necessitating thus the development of a framework that can handle a dynamics model. In a dynamics model, as opposed to a kinematics model, the control inputs are the forces (or accelerations) that act upon the system. In this paper, we provide a tractable solution to the RTL motion planning problem for dynamics models of mobile robots.

2. Problem description

We consider a mobile robot which is modeled by the second order system Σ (*dynamics model*):

$$\begin{aligned} &\ddot{x}(t) = u(t), \quad t \geq 0, \qquad x(0) \in X_0, \qquad \dot{x}(0) = 0, \\ &x(t) \in X, \qquad u(t) \in U = \{\mu \in \mathbb{R}^2 \mid \|\mu\| \leq u_{\max}\} \end{aligned} \tag{1}$$

where $x(t) \in X$ is the position of the robot in the plane, $X \subseteq \mathbb{R}^2$ is the free workspace of the robot and $X_0 \subseteq X$ is a compact set that represents the set of initial positions. Note that the robot is initially at rest, i.e., $\dot{x}(0) = 0$, and that the acceleration bound $u_{\max} > 0$ models the constraints on the control input $u(t)$ (forces or acceleration). Here, $\|\cdot\|$ is the Euclidean norm.

The goal of this paper is to construct a hybrid controller that generates control inputs $u(t)$ for system Σ so that for the set of initial states X_0, the resulting motion $x(t)$ satisfies a formula specification ϕ in the propositional temporal logic over the reals (Reynolds, 2001). Following Reynolds (2001), we refer to this logic as RTL. For the high level planning problem, we consider the existence of a number of regions of interest to the user. Such regions could be rooms and corridors in an indoor environment or areas to be surveyed in an outdoor environment. Let $\Pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$ be a finite set of symbols that label these areas. The denotation $[\![\cdot]\!] : \Pi \to \mathcal{P}(X)$ of each symbol in Π is a subset of X, i.e., for any $\pi \in \Pi$ we have $[\![\pi]\!] \subseteq X$. Here, $\mathcal{P}(\Gamma)$ denotes the powerset of a set Γ. We reserve the symbol π_0 to model the free workspace of the robot, i.e., $[\![\pi_0]\!] = X$.

In order to make apparent the use of RTL for the composition of motion planning specifications, we first give an informal description of the traditional and temporal operators. The formal syntax and semantics of RTL are presented in Section 3. RTL formulas are built over a set of propositions, the set Π in our case, using combinations of the traditional and temporal operators. Traditional logic operators are the *conjunction* ($\wedge$), *disjunction* ($\vee$) and *negation* ($\neg$). Some of the temporal operators are *eventually* ($\diamond$), *always* ($\Box$), *until* ($\mathcal{U}$) and *release* ($\mathcal{R}$). The propositional temporal logic over the reals can describe the usual properties of interest for control problems, i.e., *reachability* ($\diamond\pi$) and *safety:* ($\Box\pi$ or $\Box\neg\pi$). Beyond the usual properties, RTL can capture sequences of events and infinite behaviors. For example:

- **Reachability while avoiding regions:** The formula $\neg(\pi_1 \vee \pi_2 \vee \cdots \vee \pi_n)\mathcal{U}\pi_{n+1}$ expresses the property that the sets $[\![\pi_i]\!]$ for $i = 1, \ldots, n$ should be avoided until $[\![\pi_{n+1}]\!]$ is reached.
- **Sequencing:** The requirement that we must visit $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ in that order is captured by the formula $\diamond(\pi_1 \wedge \diamond(\pi_2 \wedge \diamond\pi_3))$.
- **Coverage:** Formula $\diamond\pi_1 \wedge \diamond\pi_2 \wedge \cdots \wedge \diamond\pi_n$ reads as the system will eventually reach $[\![\pi_1]\!]$ and eventually $[\![\pi_2]\!]$ and …eventually $[\![\pi_n]\!]$, requiring the system to eventually visit all regions of interest without imposing any ordering.
- **Recurrence (Liveness):** The formula $\Box(\diamond\pi_1 \wedge \diamond\pi_2 \wedge \cdots \wedge \diamond\pi_n)$ requires that the trajectory does whatever the coverage does and, in addition, will force the system to repeat the desired objective infinitely often.

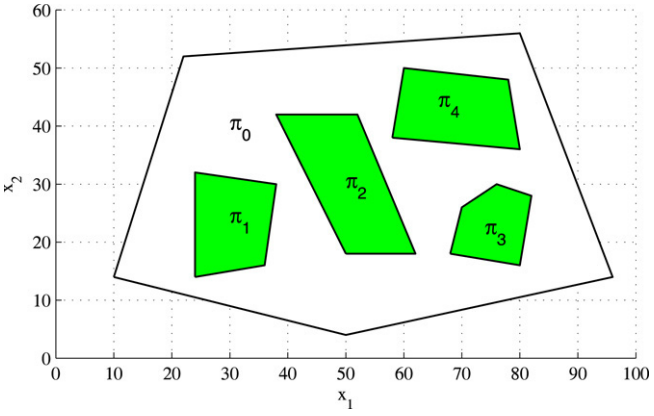

Fig. 1. The simple environment of Example 1. The four regions of interest $\pi_1, \pi_2, \pi_3, \pi_4$ are enclosed by the polygonal region labeled by π_0.

More complicated specifications can be composed from the basic specifications using the logic operators. In order to better explain the different steps in our framework, we consider throughout the paper the following example.

Example 1. Consider a robot that is moving in a convex polygonal environment π_0 with four areas of interest denoted by $\pi_1, \pi_2, \pi_3, \pi_4$ (see Fig. 1). Initially, the robot is placed somewhere in the region labeled by π_1 and its velocity is set to zero. The robot must accomplish the following task: "Visit area $[\![\pi_2]\!]$, then area $[\![\pi_3]\!]$, then area $[\![\pi_4]\!]$ and, finally, return to and stay in region $[\![\pi_1]\!]$ while avoiding areas $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$". Also, it is implied that the robot should always remain inside the free workspace X, i.e., region $[\![\pi_0]\!]$, and that $X_0 = [\![\pi_1]\!]$.

In this paper, for such specifications, we provide a computational solution to the following problem.

Problem 2. Given the system Σ and an RTL formula ϕ, construct a hybrid controller H_ϕ for Σ such that the trajectories of the closed-loop system satisfy formula ϕ.

We propose a hierarchical synthesis approach which consists of three components: tracking control using approximate simulation relations (Girard & Pappas, 2006), robust satisfaction of RTL formulas (Fainekos, Girard, & Pappas, 2007) and hybrid control for motion planning (Fainekos et al., 2005). First, Σ is abstracted to the first order system Σ' (*kinematics model*):

$$\begin{aligned} &\dot{z}(t) = v(t), \quad t \geq 0, \qquad z(0) \in Z_0, \\ &z(t) \in Z, \qquad v(t) \in V = \{\nu \in \mathbb{R}^2 \mid \|\nu\| \leq v_{\max}\} \end{aligned} \tag{2}$$

where $z(t) \in Z$ is the position of the robot in the kinematics model, $Z \subseteq \mathbb{R}^2$ is a modified free workspace, $Z_0 = X_0$ is the set of possible initial positions and $v_{\max} > 0$ is a velocity bound on the control input values $v(t)$. Using the notion of approximate simulation relation, we evaluate the precision δ with which the system Σ is able to track the trajectories of the abstraction Σ' and design a continuous tracking controller that we call *interface*. Second, from the RTL formula ϕ and the precision δ, we derive a more robust formula ϕ' such that if a trajectory z satisfies ϕ', then any trajectory x remaining at time t within distance δ from $z(t)$ satisfies formula ϕ. Thirdly, we design a hybrid controller $H'_{\phi'}$ for the abstraction Σ', so that the trajectories of the closed-loop system satisfy the formula ϕ'. Finally, by putting these three components together, as shown in Fig. 2, we design a hybrid controller H_ϕ solving Problem 2. In the following, we detail each step of our approach.

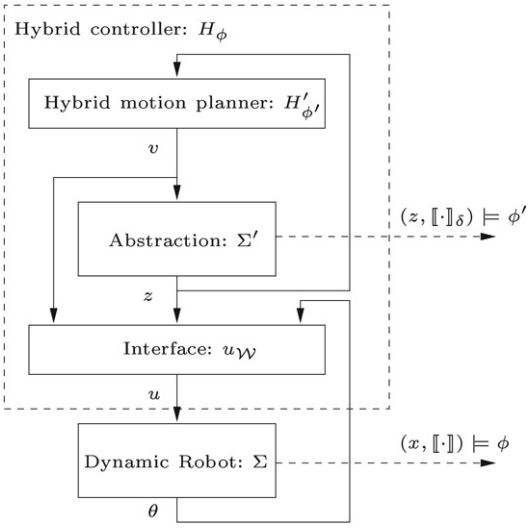

Fig. 2. Hierarchical architecture of the hybrid controller H_ϕ.

3. Propositional temporal logic over the reals

Physical processes, such as the motion of a robot, evolve in continuous time. As such, it is more intuitive for the user to state the desired robotic behavior using temporal logics with underlying continuous time semantics (Reynolds, 2001) instead of discrete ones (Pnueli, 1977). In this paper, we advocate the applicability of the propositional temporal logic over the reals with the until connective (RTL) (Reynolds, 2001) as a natural formalism for a motion planning specification language. First, we introduce the syntax of RTL formulas in Negation Normal Form (NNF) (Clarke et al., 1999, Section 9.4).

Definition 3 (*RTL/LTL Syntax*). The set Φ_Π of all well-formed formulas (wff) over the set of atomic propositions Π is constructed using the grammar

$$\phi ::= \top \mid \bot \mid \pi \mid \neg\pi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \mathcal{U} \phi \mid \phi \mathcal{R} \phi$$

where $\pi \in \Pi$ and $\top, \bot$ are the Boolean constants true and false respectively. If the rule $\neg\pi$ is dropped from the grammar, then no negation operator appears in a formula and the set of wff is denoted by Φ^+_Π.

Formally, the semantics of RTL formulas is defined over continuous time Boolean signals. Here, we instantiate the definitions of the semantics over abstractions of the trajectories of the system Σ with respect to the sets $[\![\pi]\!]$ for all $\pi \in \Pi$. Let $(x, [\![\cdot]\!]) \models \phi$ denote the satisfaction of the RTL formula ϕ over the trajectory x starting at time 0 with respect to the atomic proposition mapping $[\![\cdot]\!]$. If x does not satisfy ϕ under the map $[\![\cdot]\!]$, then we write $(x, [\![\cdot]\!]) \not\models \phi$. If all the trajectories x of the system Σ driven by a hybrid controller H and associated to an initial state in X_0 are such that $(x, [\![\cdot]\!]) \models \phi$, then we write $([\Sigma, H], [\![\cdot]\!]) \models \phi$ and we say that $[\Sigma, H]$ satisfies ϕ. In the following, given any function f from some time domain $\mathbb{T}$ (e.g., $\mathbb{R}$ or $\mathbb{N}$) to some set $\mathbb{X}$, we define $f|_t$ for $t \in \mathbb{T}$ to be the t time shift of f with definition $f|_t(t') = f(t + t')$ for $t' \in \mathbb{T}$.

Definition 4 (*RTL Semantics*). Let x be a trajectory of Σ. The semantics of any formula $\phi \in \Phi_\Pi$ can be recursively defined as:

$(x, [\![\cdot]\!]) \models \top, \quad (x, [\![\cdot]\!]) \not\models \bot$
$(x, [\![\cdot]\!]) \models \pi \quad$ iff $x(0) \in [\![\pi]\!]$
$(x, [\![\cdot]\!]) \models \neg\pi \quad$ iff $x(0) \notin [\![\pi]\!]$
$(x, [\![\cdot]\!]) \models \phi_1 \vee \phi_2 \quad$ iff $(x, [\![\cdot]\!]) \models \phi_1$ or $(x, [\![\cdot]\!]) \models \phi_2$
$(x, [\![\cdot]\!]) \models \phi_1 \wedge \phi_2 \quad$ iff $(x, [\![\cdot]\!]) \models \phi_1$ and $(x, [\![\cdot]\!]) \models \phi_2$
$(x, [\![\cdot]\!]) \models \phi_1 \mathcal{U} \phi_2 \quad$ iff $\exists t \geq 0$ such that $(x|_t, [\![\cdot]\!]) \models \phi_2$
and $\forall t'$ with $0 \leq t' < t$ we have $(x|_{t'}, [\![\cdot]\!]) \models \phi_1$
$(x, [\![\cdot]\!]) \models \phi_1 \mathcal{R} \phi_2 \quad$ iff $\forall t \geq 0$ we have $(x|_t, [\![\cdot]\!]) \models \phi_2$
or $\exists t'$ such that $0 \leq t' < t$ and $(x|_{t'}, [\![\cdot]\!]) \models \phi_1$

where $t, t' \in \mathbb{R}_{\geq 0}$.

Therefore, the formula $\phi_1 \mathcal{U} \phi_2$ intuitively expresses the property that over the trajectory x, ϕ_1 is true until ϕ_2 becomes true. The release operator $\phi_1 \mathcal{R} \phi_2$ states that ϕ_2 should always hold, a requirement which is released when ϕ_1 becomes true. Furthermore, we can derive additional temporal operators such as *eventually* $\diamond\phi = \top \mathcal{U} \phi$ and *always* $\Box\phi = \bot \mathcal{R} \phi$. The formula $\diamond\phi$ indicates that over the trajectory x the subformula ϕ eventually becomes true, whereas $\Box\phi$ indicates that ϕ is always true over x.

Example 5. Going back to Example 1, we can now formally define the specification using an RTL formula as: $\psi_1 = \Box\pi_0 \wedge \diamond(\pi_2 \wedge \diamond(\pi_3 \wedge \diamond(\pi_4 \wedge (\neg\pi_2 \wedge \neg\pi_3) \mathcal{U} \Box\pi_1)))$.

Finally, one important assumption, which we need to make when we write specifications for physical processes, is that the trajectories must satisfy the property of *finite variability* (Barringer, Kuiper, & Pnueli, 1986). The finite variability property requires that within a finite amount of time there cannot be an infinite number of changes in the satisfaction of the atomic propositions with respect to the trajectory. In other words, we should not consider Zeno trajectories (Lygeros, Johansson, Simic, Zhang, & Sastry, 2003). We address this issue in the design of our hybrid controllers in Section 6.3.

4. Tracking using approximate simulation

In this section, we present a framework for tracking control with guaranteed error bounds. It allows us to design an interface between the dynamics model Σ and its kinematics abstraction Σ' so that Σ is able to track the trajectories of Σ' with a given precision. It is based on the notion of approximate simulation relation (Girard & Pappas, 2007). Whereas exact simulation relations require the observations, i.e., $x(t)$ and $z(t)$, of two systems to be identical, approximate simulation relations allow them to be different provided their distance remains bounded by some parameter.

Let us first rewrite the second order model Σ as a system of first order differential equations

$$\Sigma : \begin{cases} \dot{x}(t) = y(t), & x(t) \in X, \quad x(0) \in X_0 \\ \dot{y}(t) = u(t), & y(t) \in \mathbb{R}^2, \quad y(0) = [0\ 0]^{\mathrm{T}} \end{cases}$$

where $x(t)$ is the position of the mobile robot and $y(t)$ its velocity at time $t \geq 0$. Here, T denotes the transpose. If we let $\theta(t) = [x^{\mathrm{T}}(t)\ y^{\mathrm{T}}(t)]^{\mathrm{T}}$, i.e., $\theta : \mathbb{R}_{\geq 0} \to \mathbb{R}^4$, with $\theta(0) \in \Theta_0 = X_0 \times \{(0, 0)\}$, then

$$\dot{\theta}(t) = A\theta(t) + Bu(t) \quad \text{and} \quad x(t) = C_x\theta(t), \qquad y(t) = C_y\theta(t)$$

where

$$A = \begin{bmatrix} \mathbf{0}_2 & \mathbf{1}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}, \qquad B = \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{1}_2 \end{bmatrix},$$
$$C_x = \begin{bmatrix} \mathbf{1}_2 & \mathbf{0}_2 \end{bmatrix}, \qquad C_y = \begin{bmatrix} \mathbf{0}_2 & \mathbf{1}_2 \end{bmatrix}.$$

Here, $\mathbf{0}_m$ and $\mathbf{1}_m$ are the zero and identity $m \times m$ matrices respectively. Then, the approximate simulation relation is defined as follows.

Definition 6 (*Simulation Relation*). A relation $\mathcal{W} \subseteq \mathbb{R}^2 \times \mathbb{R}^4$ is an approximate simulation relation of precision δ of Σ' by Σ if for all $(z_0, \theta_0) \in \mathcal{W}$,

(1) $\|z_0 - C_x\theta_0\| \leq \delta$.
(2) For all state trajectories z of Σ' such that $z(0) = z_0$ there exists a state trajectory θ of Σ such that $\theta(0) = \theta_0$ and $\forall t \geq 0$, $(z(t), \theta(t)) \in \mathcal{W}$.

An interface associated to the approximate simulation relation $\mathcal{W}$ allows us to choose the input of Σ so that the states of Σ' and Σ remain in $\mathcal{W}$.

Definition 7 (*Interface*). A continuous function $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ is an interface associated with an approximate simulation relation $\mathcal{W}$, if for all $(z_0, \theta_0) \in \mathcal{W}$, for all trajectories z of Σ' associated with a given input function v such that $z(0) = z_0$, the trajectory θ of Σ starting at $\theta(0) = \theta_0$ given by the solution of

$$\dot{\theta}(t) = A\theta(t) + Bu_{\mathcal{W}}(v(t), z(t), \theta(t)) \tag{3}$$

satisfies for all $t \geq 0$, $(z(t), \theta(t)) \in \mathcal{W}$.

Thus, by interconnecting Σ and Σ' through the interface $u_{\mathcal{W}}$ as shown on Fig. 2, the system Σ tracks the trajectories of the abstraction Σ' with precision δ. The next result is immediate from Definitions 6 and 7.

Proposition 8. *Let $\theta_0 \in \Theta_0$ and $z_0 = C_x\theta_0 \in Z_0$ such that $(z_0, \theta_0) \in \mathcal{W}$, then for all trajectories z of Σ' associated with a given input function v and initial state z_0, the trajectory θ of Σ given by (3) for $\theta(0) = \theta_0$, satisfies for all $t \geq 0$, $\|C_x\theta(t) - z(t)\| \leq \delta$.*

The construction of approximate simulation relations can be done effectively using a class of functions called simulation functions (Girard & Pappas, 2007). Essentially, a simulation function of Σ' by Σ is a positive function bounding the distance between the observations and non-increasing under the parallel evolution of the systems.

Definition 9 (*Simulation Function*). Let $\mathcal{F} : \mathbb{R}^2 \times \mathbb{R}^4 \to \mathbb{R}_{\geq 0}$ be a continuous and piecewise differentiable function and $u_{\mathcal{F}} : V \times \mathbb{R}^2 \times \mathbb{R}^4 \to \mathbb{R}^2$ be a continuous function. If for all $(z, \theta) \in \mathbb{R}^2 \times \mathbb{R}^4$ the following two inequalities hold

$$\mathcal{F}(z, \theta) \geq \|z - C_x\theta\|^2 \tag{4}$$

$$\sup_{v \in V} \left(\frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) \right) \leq 0 \tag{5}$$

then $\mathcal{F}$ is a simulation function of Σ' by Σ.

Then, approximate simulation relations can be defined as level sets of the simulation function.

Theorem 10. *Let the relation $\mathcal{W} \subseteq \mathbb{R}^2 \times \mathbb{R}^4$ be given by*

$$\mathcal{W} = \left\{ (z, \theta) \in \mathbb{R}^2 \times \mathbb{R}^4 \mid \mathcal{F}(z, \theta) \leq \delta^2 \right\}.$$

If for all $v \in V$, for all $(z, \theta) \in \mathcal{W}$, we have $u_{\mathcal{F}}(v, z, \theta) \in U$, then $\mathcal{W}$ is an approximate simulation relation of precision δ of Σ' by Σ and $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ given by $u_{\mathcal{W}}(v, z, \theta) = u_{\mathcal{F}}(v, z, \theta)$ is an associated interface.

Proof. Let $(z_0, \theta_0) \in \mathcal{W}$, then from (4), we have $\|z_0 - C_x\theta_0\| \leq \sqrt{\mathcal{F}(z_0, \theta_0)} \leq \delta$. Let z be a trajectory of Σ' generated by a given input function v such that $z(0) = z_0$. Let θ starting at $\theta(0) = \theta_0$ be given by the solution of $\dot{\theta}(t) = A\theta(t) + Bu_{\mathcal{F}}(v(t), z(t), \theta(t))$. From Eq. (5), we have that $\mathrm{d}\mathcal{F}(z(t), \theta(t))/\mathrm{d}t \leq 0$. Therefore, for all $t \geq 0$, $(z(t), \theta(t)) \in \mathcal{W}$. Furthermore, it implies that for all $t \geq 0$, $u_{\mathcal{F}}(v(t), z(t), \theta(t)) \in U$. Thus, θ is a trajectory of Σ which allows us to conclude. □

Now we are in a position to state the result that will enable us to perform tracking control.

Proposition 11. *Assume that for the systems Σ and Σ' the constraints $u_{\max}$ and $v_{\max}$ satisfy the inequality*

$$\frac{v_{\max}}{2} \left(1 + |1 - 1/\alpha| + 2/\sqrt{\alpha}\right) \leq u_{\max} \tag{6}$$

for some $\alpha > 0$. Then, an approximate simulation relation of precision $\delta = 2v_{\max}$ of Σ' by Σ is given by

$$\mathcal{W} = \{(z, \theta) \in \mathbb{R}^2 \times \mathbb{R}^4 \mid \mathcal{F}(z, \theta) \leq 4v_{\max}^2\}$$

where $\mathcal{F}(z, \theta) = \max\left(Q(z, \theta), 4v_{\max}^2\right)$ with

$$Q(z, \theta) = \|C_x\theta - z\|^2 + \alpha\|C_x\theta - z + 2C_y\theta\|^2$$

and $u_{\mathcal{W}}(v, z, \theta) = \frac{v}{2} + \frac{-1-\alpha}{4\alpha}(C_x\theta - z) - C_y\theta$ is an associated interface.

Proof. First, let us remark that (4) clearly holds. Now, let $u_{\mathcal{F}}(v, z, \theta) = u_{\mathcal{W}}(v, z, \theta)$. If $Q(z, \theta) \leq 4v_{\max}^2$, then it is clear that (5) holds. If $Q(z, \theta) \geq 4v_{\max}^2$, then we can show that (see Fainekos (2008) and Girard and Pappas (2006) for additional details)

$$\begin{aligned} &\frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) \\ &= -Q(z, \theta) - 2(C_x\theta - z) \cdot v \\ &\leq -Q(z, \theta) + 2v_{\max}\|C_x\theta - z\|. \end{aligned}$$

Since $\|C_x\theta - z\|^2 \leq Q(z, \theta)$, we have

$$\begin{aligned} &\frac{\partial \mathcal{F}(z, \theta)}{\partial z} v + \frac{\partial \mathcal{F}(z, \theta)}{\partial \theta} (A\theta + Bu_{\mathcal{F}}(v, z, \theta)) \\ &\leq -Q(z, \theta) + 2v_{\max}\sqrt{Q(z, \theta)} \\ &\leq \sqrt{Q(z, \theta)}(2v_{\max} - \sqrt{Q(z, \theta)}). \end{aligned}$$

Since $Q(z, \theta) \geq 4v_{\max}^2$, Eq. (5) holds and $\mathcal{F}$ is a simulation function of Σ' by Σ, and $u_{\mathcal{F}}$ is an associated interface. Moreover, for all $v \in V$, $(z, \theta) \in \mathcal{W}$, the interface $u_{\mathcal{F}}(v, z, \theta)$ satisfies the velocity constraints of Σ:

$$\begin{aligned} \|u_{\mathcal{F}}\| = \|u_{\mathcal{W}}\| &= \left\| \frac{v}{2} + \frac{-1 + \alpha - 2\alpha}{4\alpha}(C_x\theta - z) - C_y\theta \right\| \\ &= \left\| \frac{v}{2} + \frac{-1 + \alpha}{4\alpha}(C_x\theta - z) - \frac{1}{2}(C_x\theta - z + 2C_y\theta) \right\| \\ &\leq \frac{v_{\max}}{2} + \frac{|-1 + \alpha|}{4\alpha}\sqrt{\mathcal{F}(z, \theta)} + \frac{1}{2}\sqrt{\frac{\mathcal{F}(z, \theta)}{\alpha}} \\ &\leq \frac{v_{\max}}{2}\left(1 + |1 - 1/\alpha| + 2/\sqrt{\alpha}\right) \leq u_{\max}. \end{aligned}$$

Therefore, Theorem 10 applies and $\mathcal{W}$ is an approximate simulation relation of precision $2v_{\max}$ of Σ' by Σ and an associated interface is given by $u_{\mathcal{W}}(v, z, \theta) = u_{\mathcal{F}}(v, z, \theta)$. □

The importance of Proposition 11 is the following. Assume that the initial state of the abstraction Σ' is chosen so that $z(0) = C_x\theta(0)$ and that Σ' and Σ are interconnected through the interface $u_{\mathcal{W}}$. Then, from Proposition 8, the observed trajectories $x(t)$ of system Σ track the trajectories $z(t)$ of Σ' with precision $2v_{\max}$.

5. Robust interpretation of RTL formulas

In the previous section, we designed a control interface which enables the dynamic model Σ to track its abstract kinematic model Σ' with accuracy $2v_{\max}$. In this section, we define a new mapping $[\![\cdot]\!]_\delta$ for the atomic propositions which takes into account a bound δ on the tracking error. The new map $[\![\cdot]\!]_\delta$ provides us with a δ-robust interpretation of the motion planning specification ϕ. Intuitively, in order to achieve a robust interpretation of the specification ϕ, $[\![\cdot]\!]_\delta$ should contract by δ the areas that must be visited and expand by δ the areas that must be avoided. The fact that we have

introduced RTL syntax in NNF enables us to classify the atomic propositions in the input formula ϕ according to whether they represent regions that must be reached (no negation in front of the atomic proposition) or avoided (a negation operator appears in front of the atomic proposition).

Furthermore, for technical reasons, we need to remove any negation operators that appear in the input formula. Therefore, we introduce the *extended set of atomic propositions* Ξ_Π. In detail, we first define two new sets of symbols $\Xi_\Pi^+ = \{\xi_\pi \mid \pi \in \Pi\}$ and $\Xi_\Pi^- = \{\xi_{\neg\pi} \mid \pi \in \Pi\}$ and, then, we set $\Xi_\Pi = \Xi_\Pi^+ \cup \Xi_\Pi^-$. We also define a translation algorithm $\mathbf{pos} : \Phi_\Pi \to \Phi^+_{\Xi_\Pi}$ which takes as input an RTL formula ϕ in NNF and returns a formula $\mathbf{pos}(\phi)$ where the occurrences of the terms π and $\neg\pi$ have been replaced by the members ξ_π and $\xi_{\neg\pi}$ of Ξ_Π respectively. Since we have a new set of atomic propositions, namely Ξ_Π, we need to define a new map $[\![\cdot]\!]^\epsilon : \Xi_\Pi \to 2^X$ for the interpretation of the propositions. This is straightforward: $\forall \xi \in \Xi_\Pi$, if $\xi = \xi_\pi$, then $[\![\xi]\!]^\epsilon =: [\![\pi]\!]$, else (i.e., if $\xi = \xi_{\neg\pi}$) $[\![\xi]\!]^\epsilon =: X \setminus [\![\pi]\!]$. Then, the following result is immediate from the definition of $[\![\cdot]\!]^\epsilon$.

Lemma 12. *Given a formula $\phi \in \Phi_\Pi$, a map $[\![\cdot]\!] : \Pi \to 2^X$ and a trajectory x of Σ, we have $(x, [\![\cdot]\!]) \models \phi$ iff $(x, [\![\cdot]\!]^\epsilon) \models \mathbf{pos}(\phi)$.*

The importance of the previous lemma is the following. Since a formula $\phi \in \Phi_\Pi$ is equivalent to the formula $\phi' = \mathbf{pos}(\phi)$ under the maps $[\![\cdot]\!] : \Pi \to 2^X$ and $[\![\cdot]\!]^\epsilon : \Xi_\Pi \to 2^X$ respectively, for the rest of the paper we can assume that the input specification is given without any negation operators. That is, the next results are given with respect to a formula $\phi' \in \Phi^+_{\Xi_\Pi}$ and a map $[\![\cdot]\!]^\epsilon : \Xi_\Pi \to 2^X$. For clarity in the presentation, we denote all RTL formulas in NNF without any negation operator using primed Greek letters, e.g., ϕ', ϕ'_1, ψ'.

At this point, we have distinguished the regions that must be avoided (Ξ_Π^-) and the regions that must be reached (Ξ_Π^+). We proceed to formally defining what we mean by region contraction in order to define our notion of robustness.

Definition 13 (δ-*Contraction*). Given a radius $\delta \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$ and a point λ in a normed space Λ, the δ-*ball* centered at λ is defined as $B_\delta(\lambda) = \{\lambda' \in \Lambda \mid \|\lambda - \lambda'\| < \delta\}$. If $\Gamma \subseteq \Lambda$, then $C_\delta(\Gamma) = \{\lambda \in \Lambda \mid B_\delta(\lambda) \subseteq \Gamma\}$ is the δ-contraction of the set Γ.

Now, the δ-robust interpretation of a given RTL formula ϕ can be achieved by simply introducing a new map $[\![\cdot]\!]_\delta : \Xi_\Pi \to 2^Z$, where $Z = C_\delta(X)$ is the free workspace of Σ'. For a given $\delta \in \mathbb{R}_{\geq 0}$, the definition of the map $[\![\cdot]\!]_\delta$ is founded on the map $[\![\cdot]\!]^\epsilon$ as follows:

$$\forall \xi \in \Xi_\Pi, \qquad [\![\xi]\!]_\delta =: cl(C_\delta([\![\xi]\!]^\epsilon)).$$

The operator $cl(\Gamma)$ denotes the closure of a set Γ, that is, the intersection of all closed sets containing Γ.

Example 14. Let us revisit Examples 1 and 5. The formula ψ_1 is converted to $\psi'_1 = \mathbf{pos}(\psi_1) = \Box\xi_{\pi_0} \wedge \Diamond(\xi_{\pi_2} \wedge \Diamond(\xi_{\pi_3} \wedge \Diamond(\xi_{\pi_4} \wedge (\xi_{\neg\pi_2} \wedge \xi_{\neg\pi_3}) \mathcal{U} \Box\xi_{\pi_1})))$. We can now apply the contraction operation on the regions of interest labeled by Ξ_Π and the free workspace X and derive the δ-robust interpretation of the propositions in Ξ_Π and the modified workspace Z (see Fig. 3). For the purposes of this example, we define the map $h_\delta : Z \to 2^{\Xi_\Pi}$ such that for any $z \in Z$ we have $h_\delta(z) = \{\xi \in \Xi_\Pi \mid z \in [\![\xi]\!]_\delta\}$. Any point z in the cell 10 (*yellow* or *light gray*) is labeled by the set of propositions $h_\delta(z) = \{\xi_{\pi_0}, \xi_{\neg\pi_1}, \xi_{\pi_2}, \xi_{\neg\pi_3}, \xi_{\neg\pi_4}\}$, while any point z in the annulus region consisting of the cells 6, 7, 8 and 9 (*red* or *dark gray*) is labeled by the set $h_\delta(z) = \{\xi_{\pi_0}, \xi_{\neg\pi_1}, \xi_{\neg\pi_3}, \xi_{\neg\pi_4}\}$. Notice that $Z = [\![\xi_{\pi_0}]\!]_\delta$ and that $Z_0 = X_0 = [\![\pi_1]\!]$.

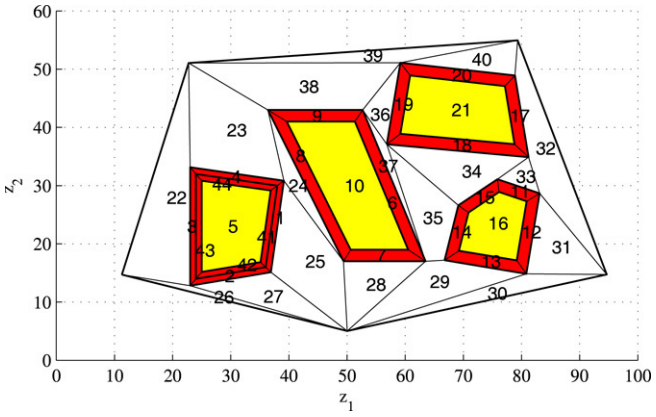

Fig. 3. A convex cell decomposition of the modified workspace of Example 14 for $\delta = 1 + \varepsilon$ (with ε sufficiently small).

Remark 15. The δ-contraction of a polyhedral set is not always a polyhedral set. In order to maintain a polyhedral description for all the sets, we under-approximate the δ-contraction by the inward δ-offset. Informally, the inward δ-offset of a polyhedral set is the inward δ-displacement of its facets along the corresponding normal directions. Since the δ-offset is an under-approximation of the δ-contraction, Theorem 16 still holds.

The following theorem is the connecting link between the specifications satisfied by the abstraction Σ' and the concrete system Σ. Informally, it states that given $\delta > 0$ if a trajectory z of Σ' satisfies the δ-robust interpretation of the input specification ϕ' and the trajectories z and x always remain δ-close, then x will satisfy the same non-robust specification ϕ'.

Theorem 16. *Consider a formula $\phi' \in \Phi^+_{\Xi_\Pi}$, a map $[\![\cdot]\!]^\epsilon : \Xi_\Pi \to 2^X$ and a number $\delta \in \mathbb{R}_{>0}$, then for all trajectories x of Σ and z of Σ' such that $\|z(t) - x(t)\| < \delta$ for all $t \geq 0$, we have $(z, [\![\cdot]\!]_\delta) \models \phi' \Longrightarrow (x, [\![\cdot]\!]^\epsilon) \models \phi'$.*

Proof. By induction on the structure of ϕ'.

Case $\phi' = \xi \in \Xi_\Pi$: We have $(z, [\![\cdot]\!]_\delta) \models \xi$ iff $z(0) \in [\![\xi]\!]_\delta = C_\delta([\![\xi]\!]^\epsilon)$. The later implies that $B_\delta(z(0)) \subseteq [\![\xi]\!]^\epsilon$. Since $\|z(0) - x(0)\| < \delta$, we immediately get that $x(0) \in [\![\xi]\!]^\epsilon$. Thus, $(x, [\![\cdot]\!]^\epsilon) \models \xi$.

Case $\phi' = \phi'_1 \mathcal{U} \phi'_2$: We have $(z, [\![\cdot]\!]_\delta) \models \phi'_1 \mathcal{U} \phi'_2$ iff by definition there exists $t \geq 0$ such that $(z|_t, [\![\cdot]\!]_\delta) \models \phi'_2$ and for all $t' \in [0, t)$ we have $(z|_{t'}, [\![\cdot]\!]_\delta) \models \phi'_1$. Since $\|z(t'') - x(t'')\| < \delta$ for all $t'' \geq 0$, by the induction hypothesis, we get that $(x|_t, [\![\cdot]\!]^\epsilon) \models \phi'_2$ and that for all $t' \in [0, t)$, $(x|_{t'}, [\![\cdot]\!]^\epsilon) \models \phi'_1$. Therefore, $(x, [\![\cdot]\!]^\epsilon) \models \phi'_1 \mathcal{U} \phi'_2$.

The other cases (see Fainekos (2008)) are either similar (release) or straightforward (conjunction and disjunction). □

6. Temporal logic motion planning

Having presented the connection between the dynamics model Σ and the kinematics model Σ', we proceed to solving the temporal logic motion planning problem for Σ'. Formally, we solve the following more general problem.

Problem 17. Given the system Σ', a set of atomic propositions Ξ_Π, an RTL formula $\phi' \in \Phi^+_{\Xi_\Pi}$ and a map $[\![\cdot]\!]_\delta : \Xi_\Pi \to 2^Z$, construct a hybrid controller $H'_{\phi'}$ such that $([\Sigma', H'_{\phi'}], [\![\cdot]\!]_\delta) \models \phi'$.

Our solution consists of the following three steps: (1) reduction of the continuous environment Z into a discrete graph, (2) temporal logic planning over the discrete graph, and (3) continuous implementation of the final discrete plan.

6.1. Discrete abstraction of robot motion

In order to use discrete logics to reason about continuous systems, we need to construct a finite partition of the continuous state space Z with respect to the map $[\![\cdot]\!]_\delta$. For that purpose, we can use many efficient cell decomposition methods for polygonal environments (Choset et al., 2005; LaValle, 2006). Note that by employing any workspace decomposition method we can actually construct a *topological graph* (or *roadmap* as it is sometimes referred to LaValle (2006)). A topological graph describes which cells are topologically adjacent, i.e., each node in the graph represents a cell and each edge in the graph implies topological adjacency of the cells. No matter which decomposition algorithm is employed, the workspace's decomposition must be *proposition preserving* with respect to the mapping $[\![\cdot]\!]_\delta$. In other words, we require that all the points which belong to the same cell must be labeled by the same set of propositions.

On a more formal note, we can partition the workspace Z with respect to the map $[\![\cdot]\!]_\delta$ and the set of initial conditions Z_0 into a number of equivalence classes, that is, into a number of sets of points which satisfy the same property (in this case the same set of propositions). Note that mathematically the set of equivalence classes consists of the interior of the cells, the edges and the vertices. In the following, we define $\mathcal{Q} = \{q_1, \ldots, q_n\}$ to be the set of all equivalence classes. Let us introduce the map $T : Z \to \mathcal{Q}$ which sends each state $z \in Z$ to one equivalence class in $\mathcal{Q}$. Then, we can formally state the proposition preserving property as follows: $\forall z_i, z_j \in Z.\, T(z_i) = T(z_j)$ implies $h_\delta(z_i) = h_\delta(z_j)$. Recall that $h_\delta(z) = \{\xi \in \Xi_\Pi \mid z \in [\![\xi]\!]_\delta\}$. Moreover, we define the "inverse" map $T^{-1} : \mathcal{Q} \to \mathcal{P}(Z)$ of T such that $T^{-1}(q)$ maps to all states $z \in Z$ which are contained in the equivalence class q.

In the following paragraphs, we present how the topological graph resulting from the decomposition of Z with respect to $[\![\cdot]\!]_\delta$ can be converted into a Finite Transition System (FTS) that serves as an abstract model of the robot motion. Posing the topological graph as an FTS allows us to use standard automata theoretic techniques (Clarke et al., 1999) in order to solve the high level planning problem.

Definition 18 (*FTS*). A Finite Transition System is a tuple $\mathcal{D} = (Q, Q_0, \to_{\mathcal{D}}, h_{\mathcal{D}}, \Xi_\Pi)$ where:

- Q is a set of states. Here, $Q \subseteq \mathcal{Q}$ is the set of equivalence classes that represent the interior of the cells.
- $Q_0 \subseteq Q$ is the set of possible initial cells. Here, Q_0 satisfies $\cup_{q_0 \in Q_0} cl(T^{-1}(q_0)) = Z_0$.
- $\to_{\mathcal{D}} \subseteq Q \times Q$ captures the topological relationship between the cells. There is a transition from cell q_i to cell q_j written as $q_i \to_{\mathcal{D}} q_j$ if the cells labeled by q_i, q_j are adjacent, i.e., $cl(T^{-1}(q_i))$ and $cl(T^{-1}(q_j))$ share a common edge. Finally, for all $q \in Q$ we add a self-loop, i.e., $q \to_{\mathcal{D}} q$.
- $h_{\mathcal{D}} : Q \to \mathcal{P}(\Xi_\Pi)$ is a map defined as $h_{\mathcal{D}}(q) = \{\xi \in \Xi_\Pi \mid T^{-1}(q) \subseteq [\![\xi]\!]_\delta\}$.

We define a *path* on the FTS to be a sequence of states (cells) and a *trace* to be the corresponding sequence of sets of propositions. Formally, a path is a function $p : \mathbb{N} \to Q$ such that for each $i \in \mathbb{N}$ we have $p(i) \to_{\mathcal{D}} p(i+1)$ and the corresponding trace is the function composition $\bar{p} = h_{\mathcal{D}} \circ p : \mathbb{N} \to \mathcal{P}(\Xi_\Pi)$.

Example 19. Let us consider the convex decomposition of the workspace of Example 14 which appears in Fig. 3. The topological graph contains 40 nodes and 73 edges. For the following examples, we let $\mathcal{D}_1$ denote the FTS which corresponds to the aforementioned topological graph.

6.2. Linear temporal logic planning

The transition system $\mathcal{D}$, which was constructed in the previous section, will serve as an abstract model of the robot's motion. We must now lift our problem formulation from the continuous to the discrete domain. For that purpose we introduce and use Linear Temporal Logic (LTL) (Pnueli, 1977) which has exactly the same syntax as RTL, but its semantics is interpreted over discrete paths generated by a finite transition system. In the following, we let $\bar{p} \models \phi$ denote the satisfiability of an LTL formula ϕ over a trace $\bar{p}$.

Definition 20 (*Discrete LTL Semantics*). The semantics of any LTL formula $\phi' \in \Phi^+_{\Xi_\Pi}$ is defined as:

$\bar{p} \models \top, \quad \bar{p} \not\models \bot, \quad \bar{p} \models \xi \quad \text{iff } \xi \in \bar{p}(0)$
$\bar{p} \models \phi'_1 \wedge \phi'_2 \quad \text{if } \bar{p} \models \phi'_1 \text{ and } \bar{p} \models \phi'_2$
$\bar{p} \models \phi'_1 \vee \phi'_2 \quad \text{if } \bar{p} \models \phi'_1 \text{ or } \bar{p} \models \phi'_2$
$\bar{p} \models \phi'_1 \mathcal{U} \phi'_2 \quad \text{if there exists } i \geq 0 \text{ such that } \bar{p}|_i \models \phi'_2$
and for all j with $0 \leq j < i$ we have $\bar{p}|_j \models \phi'_1$
$\bar{p} \models \phi'_1 \mathcal{R} \phi'_2 \quad \text{if for all } i \geq 0 \text{ we have } \bar{p}|_i \models \phi'_2$
or there exists $j \in [0, i)$ such that $\bar{p}|_j \models \phi'_1$

where $i, j \in \mathbb{N}$.

In this work, we are interested in the construction of automata that only accept the traces of $\mathcal{D}$ which satisfy the LTL formula ϕ'. Such automata (which are referred to as Büchi automata (Clarke et al., 1999, Section 9.1)) differ from the classic finite automata in that they accept infinite strings (traces of $\mathcal{D}$ in our case).

Definition 21 (*Automaton*). A Büchi automaton is a tuple $\mathcal{B} = (S_{\mathcal{B}}, s_{0\mathcal{B}}, \Omega, \lambda_{\mathcal{B}}, F_{\mathcal{B}})$ where:

- $S_{\mathcal{B}}$ is a finite set of states and $s_{0\mathcal{B}}$ is the initial state.
- Ω is an input alphabet.
- $\lambda_{\mathcal{B}} : S_{\mathcal{B}} \times \Omega \to \mathcal{P}(S_{\mathcal{B}})$ is a transition relation.
- $F_{\mathcal{B}} \subseteq S_{\mathcal{B}}$ is a set of accepting states.

In order to define what it means for a Büchi automaton to accept a trace, we must first introduce some terminology. A *run* r of $\mathcal{B}$ is a sequence of states $r : \mathbb{N} \to S_{\mathcal{B}}$ that occurs under an input trace $\bar{p}$, that is for $i = 0$ we have $r(0) = s_{0\mathcal{B}}$ and for all $i \geq 0$ we have $r(i+1) \in \lambda_{\mathcal{B}}(r(i), \bar{p}(i))$. Let $\lim(\cdot)$ be the function that returns the set of states that are encountered infinitely often in the run r of $\mathcal{B}$. Then, a run r of a Büchi automaton $\mathcal{B}$ over an infinite trace $\bar{p}$ is *accepting* if and only if $\lim(r) \cap F_{\mathcal{B}} \neq \emptyset$. Informally, a run r is accepting when some accepting state $s \in F_{\mathcal{B}}$ appears in r infinitely often. Finally, we define the language $\mathcal{L}(\mathcal{B})$ of $\mathcal{B}$ to be the set of all traces $\bar{p}$ that have a run that is accepted by $\mathcal{B}$. For each LTL formula ϕ', we can construct a Büchi automaton $\mathcal{B}_{\phi'} = (S_{\mathcal{B}_{\phi'}}, s_{0\mathcal{B}_{\phi'}}, \mathcal{P}(\Xi_\Pi), \lambda_{\mathcal{B}_{\phi'}}, F_{\mathcal{B}_{\phi'}})$ that accepts the infinite traces which satisfy the specification ϕ', i.e., $\bar{p} \in \mathcal{L}(\mathcal{B}_{\phi'})$ iff $\bar{p} \models \phi'$. The translation from an LTL formula ϕ' to a Büchi automaton $\mathcal{B}'_\phi$ is a well-studied problem and, thus, we refer the reader to Clarke et al. (1999, Section 9.4) and the references therein for the theoretical details behind this translation.

We can now use the abstract representation of the robot's motion, that is the FTS, in order to reason about the desired motion of the robot. First, we convert the FTS $\mathcal{D}$ into a Büchi automaton $\mathcal{D}'$. The translation from $\mathcal{D}$ to $\mathcal{D}'$ enables us to use standard tools and techniques from automata theory (Clarke et al., 1999, Section 9) alleviating, thus, the need for developing new theories. Translating an FTS into an automaton is a standard procedure which can be found in any formal verification textbook (see Clarke et al. (1999, Section 9.2)).

Definition 22 (*FTS to Automaton*). The Büchi automaton $\mathcal{D}'$ which corresponds to the FTS $\mathcal{D}$ is the automaton $\mathcal{D}' = (Q', q_d, 2^{(\Sigma_\Pi)}, \lambda_{\mathcal{D}'}, F_{\mathcal{D}'})$ where:

- $Q' = Q \cup \{q_d\}$ for $q_d \notin Q$.
- $\lambda_{\mathcal{D}'} : Q' \times 2^{(\Sigma_\Pi)} \to 2^{Q'}$ is the transition relation defined as: $q_j \in \lambda_{\mathcal{D}'}(q_i, l)$ iff $q_i \to_{\mathcal{D}} q_j$ and $l = h_{\mathcal{D}}(q_j)$ and $q_0 \in \lambda_{\mathcal{D}'}(q_d, l)$ iff $q_0 \in Q_0$ and $l = h_{\mathcal{D}}(q_0)$.
- $F_{\mathcal{D}'} = Q'$ is the set of accepting states.

Now that all the related terminology is defined, we can give an overview of the basic steps involved in the temporal logic planning (De Giacomo & Vardi, 1999). Our goal in this section is to generate paths on $\mathcal{D}$ that satisfy the specification ϕ'. In automata theoretic terms, we want to find the subset of the language $\mathcal{L}(\mathcal{D}')$ which also belongs to the language $\mathcal{L}(B_{\phi'})$. This subset is simply the intersection of the two languages $\mathcal{L}(\mathcal{D}') \cap \mathcal{L}(B_{\phi'})$ and it can be constructed by taking the product $\mathcal{D}' \times B_{\phi'}$ of the Büchi automaton $\mathcal{D}'$ and the Büchi automaton $B_{\phi'}$. Informally, the Büchi automaton $B_{\phi'}$ restricts the behavior of the system $\mathcal{D}'$ by permitting only certain acceptable transitions. Then, given an initial state in the FTS $\mathcal{D}$, which is an abstraction of the actual initial position of the robot, we can choose a particular trace from $\mathcal{L}(\mathcal{D}) \cap \mathcal{L}(B_{\phi'})$ according to a preferred criterion. In the following, we present the details of this construction.

Definition 23. The product automaton $\mathcal{A} = \mathcal{D}' \times B_{\phi'}$ is the automaton $\mathcal{A} = (S_{\mathcal{A}}, s_{0\mathcal{A}}, 2^{(\Sigma_\Pi)}, \lambda_{\mathcal{A}}, F_{\mathcal{A}})$ where:

- $S_{\mathcal{A}} = Q' \times S_{\mathcal{B}_{\phi'}}$ and $s_{0\mathcal{A}} = \{(q_d, s_{0\mathcal{B}_{\phi'}})\}$.
- $\lambda_{\mathcal{A}} : S_{\mathcal{A}} \times 2^{(\Sigma_\Pi)} \to 2^{(S_{\mathcal{A}})}$ such that $(q_j, s_j) \in \lambda_{\mathcal{A}}((q_i, s_i), l)$ iff $q_j \in \lambda_{\mathcal{D}'}(q_i, l)$ and $s_j \in \lambda_{B_{\phi'}}(s_i, l)$.
- $F_{\mathcal{A}} = Q' \times F$ is the set of accepting states.

By construction, the following lemma is satisfied (recall that $\bar{p}$ is a trace of $\mathcal{D}$ if and only if $\bar{p}$ is accepted by $\mathcal{D}'$).

Lemma 24 (*Adapted from De Giacomo and Vardi (1999)*). *A trace $\bar{p}$ of $\mathcal{D}$ that satisfies the specification ϕ' exists iff the language of $\mathcal{A}$ is non-empty, i.e., $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D}') \cap \mathcal{L}(B_{\phi'}) \neq \emptyset$.*

Checking the emptiness of language $\mathcal{L}(\mathcal{A})$ is an easy algorithmic problem (Clarke et al., 1999, Section 9.3). First, we convert automaton $\mathcal{A}$ to a directed graph and, then, we find the strongly connected components (SCC) (Cormen, Leiserson, Rivest, & Stein, 2001, Section 22.5) in that graph. If at least one SCC that contains an accepting state is reachable from $s_{0\mathcal{A}}$, then the language $\mathcal{L}(\mathcal{A})$ is not empty. However, we are not just interested in figuring out whether $\mathcal{L}(\mathcal{A}) = \emptyset$. We need to construct an accepting run of $\mathcal{A}$ and from that derive a discretized path for the robot on $\mathcal{D}$. The good news is that if $\mathcal{L}(\mathcal{A})$ is non-empty, then there exist accepting (infinite) runs on $\mathcal{A}$ that have a finite representation. Each such run consists of two parts. The first part is a *finite* sequence of states $r(0)r(1)\ldots r(m_f)$ which corresponds to the sequence of states starting from $r(0) = s_{0\mathcal{A}}$ and reaching a state $r(m_f) \in F_{\mathcal{A}}$. The second part is a *periodic* sequence of states $r(m_f)r(m_f+1)\ldots r(m_f+m_l)$ such that $r(m_f+m_l) = r(m_f)$ which corresponds to the part of the run that traverses some part of the strongly connected component. Here, $m_f, m_l \geq 0$ is less than or equal to the number of states in $\mathcal{D}$, i.e., $m_f, m_l \leq |Q|$.

Since in this paper we are concerned with a path planning application, it is desirable to choose an accepting run that traverses as few different states on $\mathcal{D}$ as possible. The high level description of the algorithm is as follows. First, we find all the shortest sequences of states from $s_{0\mathcal{A}}$ to all the accepting states in $F_{\mathcal{A}}$ using Breadth First Search (BFS) (Cormen et al., 2001, Section 22.2). Then, from each reachable accepting state $q_a \in F_{\mathcal{A}}$ we initiate a new BFS in order to find the shortest sequence of states that leads back to q_a. Note that if no accepting state is reachable from $s_{0\mathcal{A}}$ or no infinite loop can be found, then the language $\mathcal{L}(\mathcal{A})$ is empty and, hence, the temporal logic planning problem does not have a solution. Moreover, if $\mathcal{L}(\mathcal{A}) \neq \emptyset$, then this algorithm can potentially return a set R of accepting runs r each leading to a different accepting state in $F_{\mathcal{A}}$ with a different periodic part. From the set of runs R, we can easily derive a corresponding set of paths P on $\mathcal{D}$ such that for all $p \in P$ we have that the trace $\bar{p}$ satisfies ϕ'. The following is immediate from the definitions.

Proposition 25. *Let $pr : S_{\mathcal{A}} \to Q$ be a projection function such that $pr(q, s) = q$. If r is an accepting run of $\mathcal{A}$, then $p = (pr \circ r)|_1$ is a path on $\mathcal{D}$ such that $\bar{p} \models \phi'$.*

Any path $p \in P$ can be characterized by a pair of sequences of states (p^f, p^l). Here, $p^f = p^f_1 p^f_2 \ldots p^f_{n_f}$ denotes the finite part of the path and $p^l = p^l_1 p^l_2 \ldots p^l_{n_l}$ the periodic part (infinite loop) such that $p^f_{n_f} = p^l_1$. The relation between the pair (p^f, p^l) and the path p is given by $p(i) = p^f_{i+1}$ for $0 \leq i \leq n_f - 2$ and $p(i) = p^l_j$ with $j = ((i - n_f + 1) \bmod n_l) + 1$ for $i \geq n_f - 1$.

Example 26. The Büchi automaton $\mathcal{B}_{\psi'_1}$ that accepts the paths that satisfy ψ'_1 has 5 states (one accepting) and 13 transitions. For the conversion from LTL to Büchi automata, we use the python toolbox LTL2NBA by Fritz and Teegen, which is based on Fritz (2003). The product automaton $\mathcal{A}_1 = \mathcal{D}'_1 \times \mathcal{B}_{\psi'_1}$ has 205 states. The shortest path on the topological graph starting from cell 5 is: $(p^f, p^l) = (\{5, 41, 1, 25, 24, 8, 10, 6, 37, 35, 14, 16, 15, 34, 18, 21, 19, 36, 38, 23, 4, 44, 5\}, \{5\})$. Using Fig. 3, the reader can verify that this sequence satisfies ψ'_1 under the map $[\![\cdot]\!]_\delta$.

6.3. Continuous implementation of discrete trajectory

Our next task is to utilize each discrete path $p \in P$ in order to construct a hybrid control input $v(t)$ for $t \geq 0$ which will drive Σ' so that its trajectories $z(t)$ satisfy the RTL formula ϕ'. We achieve this desired goal by simulating (or implementing) at the continuous level each discrete transition of p. This means that if the discrete system D makes a transition $q_i \to_D q_j$, then the continuous system Σ' must match this discrete step by moving from any position in the cell $cl(T^{-1}(q_i))$ to a position in the cell $cl(T^{-1}(q_j))$. Moreover, if the periodic part in the path p consists of just a single state q_l, then we have to guarantee that the position of the robot always remains in the invariant set $T^{-1}(q_l)$. These basic control specifications imply that we need at least two types of continuous feedback control laws. We refer to these control laws as *reachability* and *cell invariant* controllers. Informally, a reachability controller drives each state inside a cell q to a predefined region on the cell's boundary, while the cell invariant controller guarantees that all the trajectories that start inside a cell q always remain in that cell.

Let us assume that we are given or that we can construct a finite collection of continuous feedback control laws $\{g_\kappa\}_{\kappa \in K}$ indexed by a control alphabet K such that for any $\kappa \in K$ we have $g_\kappa : Z_\kappa \to V$ with $Z_\kappa \subseteq Z$. In our setting, we make the following additional assumptions. First, we define the operational range of each controller to be one of the cells in the workspace of the robot, i.e., for any $\kappa \in K$ there exists some $q \in Q$ such that $Z_\kappa = cl(T^{-1}(q))$. Second, if g_κ is a reachability controller, then we require that all the trajectories which start in Z_κ must converge on the same subset of the boundary of Z_κ within finite time while never exiting Z_κ before that time. Finally, if g_κ is a cell invariant controller, then we require the all the trajectories which initiate from a point in Z_κ converge on the barycenter b_κ of Z_κ. Examples of such feedback control laws for Σ' appear in Fig. 4. A formal presentation of these types of controllers is beyond the scope of

Fig. 4. (a) Reachability and (b) Cell invariant controller.

this paper and the interested reader can find further details in Belta et al. (2005), Conner et al. (2003) and Lindemann and LaValle (2005).

The way we can compose such controllers given the pair (p^f, p^l), which characterizes a path $p \in P$, is as follows. First note that it is possible to get a finite repetition of states in the path p, for example there can exist some $i \geq 0$ such that $p(i) = p(i+1)$ but $p(i+1) \neq p(i+2)$. This situation might occur because we have introduced self-loops in the automaton D' in conjunction with the possibility that the Büchi automaton $B_{\phi'}$ might not be optimal (in the sense of number of states and transitions). Therefore, we first remove finite repetitions of states from p. Removing such repeated states from p does not change the fact that $\bar{p} \models \phi'$. This is possible because LTL formulas without the *next* time operator are *stutter invariant* (Clarke et al., 1999, Section 10). Next, we define the control alphabet to be $K = K^f \cup K^l \subseteq Q \times Q$ where $K^f = \cup_{i=1}^{n_f-1}\{(p_i^f, p_{i+1}^f)\} \cup \{(p_{n_f}^f, p_1^l)\}$ and $K^l = \cup_{i=1}^{n_l-1}\{(p_i^l, p_{i+1}^l)\} \cup \{(p_{n_l}^l, p_1^l)\}$ when $n_l > 1$ or $K^l = \emptyset$ otherwise. For any $\kappa = (q_i, q_j) \in K \setminus \{(p_{n_f}^f, p_1^l)\}$, we design g_κ to be a reachability controller that drives all initial states in $Z_\kappa = cl(T^{-1}(q_i))$ to the common edge $cl(T^{-1}(q_i)) \cap cl(T^{-1}(q_j))$. Finally for $\kappa = (p_{n_f}^f, p_1^l)$, we let g_κ be a cell invariant controller for the cell p_1^l.

It is easy to see now how we can use each pair (p^f, p^l) in order to construct a hybrid controller $H'_{\phi'}$. Starting anywhere in the cell $cl(T^{-1}(p_1^f))$, we apply the control law $g_{(p_1^f, p_2^f)}$ until the robot crosses the edge $cl(T^{-1}(p_1^f)) \cap cl(T^{-1}(p_2^f))$. At that point, we switch the control law to $g_{(p_2^f, p_3^f)}$. The above procedure is repeated until the last cell of the finite path p^f at which point we apply the cell invariant controller $g_{(p_{n_f}^f, p_1^l)}$. If the periodic part p^l of the path has only one state, i.e., $n_l = 1$, then this completes the construction of the hybrid controller $H'_{\phi'}$. If on the other hand $n_l > 1$, then we check whether the trajectory $z(t)$ has entered an ε-neighborhood of the barycenter of the cell invariant controller. If so, we apply ad infinitum the sequential composition of the controllers that correspond to the periodic part of the path p^l followed by the cell invariant controller $g_{(p_{n_f}^f, p_1^l)}$. The cell invariant controller is necessary in order to avoid Zeno behavior (Lygeros et al., 2003). Since there can only exist at most one Zeno cycle in the final hybrid automaton and this cycle is guaranteed not to generate Zeno behaviors due to the existence of the cell invariant controller, the following proposition is immediate.

Proposition 27. *The trajectories z of the system $[\Sigma', H'_{\phi'}]$ satisfy the finite variability property.*

Assuming now that Σ' is controlled by the hybrid controller $H'_{\phi'}$ which is constructed as described above, we can prove the following theorem.

Fig. 5. A trajectory of system Σ' for the path of Example 26 using the potential field controllers of Conner et al. (2003).

Theorem 28. *Let $\phi' \in \Phi^+_{\Xi\Pi}$, P be a set of paths on D such that $\forall p \in P$, we have $\bar{p} \models \phi'$ and $H'_{\phi'}$ be the corresponding hybrid controller, then $([\Sigma', H'_{\phi'}], [\![\cdot]\!]_\delta) \models \phi'$.*

Proof. For any $p \in P$ we prove that if $\bar{p} \models \phi'$, then $(z, [\![\cdot]\!]_\delta) \models \phi'$ for any trajectory z of the system $[\Sigma', H'_{\phi'}]$ starting at any $z(0) \in cl(T^{-1}(p(0)))$. The proof uses induction on the structure of ϕ'. Here, we only present two cases. The other cases are similar (see Fainekos (2008)).

Case $\bar{p} \models \xi$: Since $z(0) \in cl(T^{-1}(p(0)))$ we get that $z(0) \in [\![\xi]\!]_\delta$ (by def. $[\![\xi]\!]_\delta = cl([\![\xi]\!]_\delta)$). Hence, $(z, [\![\cdot]\!]_\delta) \models \xi$.

Case $\bar{p} \models \phi'_1 \mathcal{U} \phi'_2$: Then there exists some $i \geq 0$ such that $\bar{p}|_i \models \phi'_2$ and for all $j \in [0, i)$ we get that $\bar{p}|_j \models \phi'_1$. Consider now the trajectory z that is generated by Σ' using the controller $H'_{\phi'}$ that corresponds to the path p. The initial condition for the trajectory z is any point in the initial cell, i.e., $z(0) \in cl(T^{-1}(p(0)))$. By construction, there exists a sequence of times $0 = \tau_0 \leq \tau_1 \leq \cdots \leq \tau_i$, where τ_j for $j \in (0, i]$ is the time that the trajectory z crosses the edge $cl(T^{-1}(p(j-1))) \cap cl(T^{-1}(p(j)))$. Consider any time instant $t' \in [\tau_j, \tau_{j+1}]$ for any $j \in [0, i)$. Then, we know that $z|_{t'}(0) \in cl(T^{-1}(p|_j(0)))$. Now the induction hypothesis applies and we get that $(z|_{t'}, [\![\cdot]\!]_\delta) \models \phi'_1$. Therefore, for all $t' \in [0, \tau_i]$, we have $(z|_{t'}, [\![\cdot]\!]_\delta) \models \phi'_1$. Now, note that $z(\tau_i) \in cl(T^{-1}(p(i-1))) \cap cl(T^{-1}(p(i)))$, hence by the induction hypothesis we get that $(z|_{\tau_i}, [\![\cdot]\!]_\delta) \models \phi'_2$. Thus, if we set $t' = \tau_i$, then we are done and $(z, [\![\cdot]\!]_\delta) \models \phi'_1 \mathcal{U} \phi'_2$. Note that if $\bar{p} \models \phi'_2$, then we are immediately done since the induction hypothesis applies directly. □

Theorem 28 concludes our proposed solution to Problem 17. The following example illustrates the theoretical results presented in Section 6.

Example 29. For the construction of the hybrid controller $H'_{\phi'}$ based on the path of Example 26, we deploy the potential field controllers of Conner et al. (2003) on the cellular decomposition of Fig. 3. The resulting trajectory with initial position (35, 20) and velocity bound $v_{\max} = 0.5$ appears in Fig. 5.

7. Putting everything together

At this point, we have presented all the pieces that comprise our proposed solution to Problem 2. Now we are in a position to put all the parts together according to the hierarchy proposed in Fig. 2. The following theorem which is immediate from Proposition 11, Lemma 12 and Theorems 16 and 28 states the main result of the paper.