# Optimal Control of Nonlinear Systems with Temporal Logic Specifications

Eric M. Wolff and Richard M. Murray

**Abstract** We present a mathematical programming-based method for optimal control of nonlinear systems subject to temporal logic task specifications. We specify tasks using a fragment of linear temporal logic (LTL) that allows both finite- and infinite-horizon properties to be specified, including tasks such as surveillance, periodic motion, repeated assembly, and environmental monitoring. Our method directly encodes an LTL formula as mixed-integer linear constraints on the system variables, avoiding the computationally expensive process of creating a finite abstraction. Our approach is efficient; for common tasks our formulation uses significantly fewer binary variables than related approaches and gives the tightest possible convex relaxation. We apply our method on piecewise affine systems and certain classes of differentially flat systems. In numerical experiments, we solve temporal logic motion planning tasks for high-dimensional (10+ continuous state) systems.

## 1 Introduction

In safety-critical robotics applications involving autonomous ground and air vehicles, it is desirable to unambiguously specify the desired system behavior and automatically synthesize a controller that provably implements this behavior. Additionally, autonomous systems often have high-dimensional, nonlinear dynamics and require high-performance (not just feasible) controllers.

Linear temporal logic (LTL) is an expressive task-specification language for specifying a variety of tasks such as responding to the environment, visiting goals, periodically monitoring areas, staying safe, and remaining stable. These properties generalize classical point-to-point motion planning. Also, the widespread use of

Eric M. Wolff
California Institute of Technology, Pasadena, CA e-mail: `ewolff@caltech.edu`

Richard M. Murray
California Institute of Technology, Pasadena, CA e-mail: `murray@cds.caltech.edu`

LTL in software verification [3] makes it appealing as a common language for reasoning about the software and dynamics of autonomous systems.

Standard methods for motion planning with LTL task specifications first create a finite abstraction of the original dynamical system. This abstraction can informally be viewed as a labeled graph that represents possible behaviors of the system. Approximate finite abstractions can be computed using either sampling-based methods (e.g., RRTs) [7, 15, 18] or reachability-based approaches [2, 4, 11, 17, 28].

Given a finite abstraction of a dynamical system and an LTL specification, controllers can be automatically constructed using an automata-based approach [3, 7, 10, 15, 17]. This approach first transforms the LTL formula into an equivalent Büchi automaton whose size may be exponential in the length of the formula [3]. A product automaton is created from the finite abstraction and the Büchi automaton, and then a controller is found by graph search in the product automaton.

The main drawback of this approach is that it is expensive to compute a finite abstraction. The product automaton might also be quite large due to the size of the abstraction and the Büchi automaton. Finally, although optimal controllers can be computed for the discrete abstraction [22, 27], optimality is only with respect to the abstraction's level of refinement or asymptotic [15].

Instead of the automata-based approach, we directly encode a large class of temporal logic formulas as mixed-integer linear constraints on the original dynamical system. These constraints enforce that an infinite sequence of system states satisfies a task specification. A key component of our formulation is enforcing that the system is in a (non-convex) region at a given time. We introduce an alternative formulation for this that gives a tighter convex relaxation than the commonly used big-M approach. Our approach applies to any deterministic system model that is amenable to finite-dimensional optimization, as the temporal logic constraints are independent of any particular system dynamics or cost functions. We specifically investigate Mixed Logical Dynamic (MLD) systems [5] and certain differentially flat systems [20], whose dynamics can be encoded with mixed-integer linear constraints. MLD systems include constrained linear systems, linear hybrid automata, and piecewise affine systems. Differentially flat systems include quadrotors and car-like vehicles.

It is well-known that mixed-integer linear programming can be used for reasoning about propositional logic [8, 12], generating state-constrained trajectories [9, 21, 24], and modeling vehicle routing problems [14, 23]. The work most similar to ours is Karaman et al. [16], who consider controller synthesis for MLD systems subject to finite-horizon LTL specifications. However, finite-horizon properties are too restrictive to model a large class of interesting robotics problems, including persistent surveillance, repeated assembly, periodic motion, and environmental monitoring. Our work specifically addresses these types of periodic tasks with a novel mixed-integer formulation.

Our main contributions are 1) a novel method for encoding both finite- and infinite-horizon temporal logic properties as mixed-integer linear constraints on a system and 2) an improved encoding that has a tighter convex relaxation and uses significantly fewer binary variables for common tasks than related work [16]. The fragment of temporal logic that we consider allows one to specify properties such as safety, stability, liveness, guarantee, and response. We demonstrate how this mixed-integer programming formulation can be used with off-the-shelf optimization solvers (e.g. CPLEX [1]) to compute both feasible and optimal controllers for high-dimensional systems with temporal logic specifications.

## 2 Preliminaries

An *atomic proposition* is a statement that is *True* or *False*. A *propositional formula* is composed of only atomic propositions and propositional connectives, i.e., $\wedge$ (and), $\vee$ (or), and $\neg$ (not). Let $\mathcal{T} = \{0,1,2,\dots,T\} \subset \mathbb{N}$ denote a bounded set of discrete time instances and $\mathcal{T}^\infty = \{0,1,2,\dots\}$ denote an unbounded set of discrete time instances.

### 2.1 System model

We consider discrete-time nonlinear systems of the form

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

where $t \in \mathcal{T}^\infty$, $x \in \mathcal{X} \subseteq \mathbb{R}^{n_c} \times \{0,1\}^{n_l}$ are the continuous and binary states, $u \in \mathcal{U} \subseteq \mathbb{R}^{m_c} \times \{0,1\}^{m_l}$ are the inputs, and $x(0) = x_0 \in \mathcal{X}$ is the initial state. We assume that the system is deterministic, i.e., an initial state $x_0$ and a control input sequence $u = u_0u_1u_2\dots$ produces a unique trajectory (or run) $x = x(x_0,u) = x_0x_1x_2\dots$.

Let $AP$ be a finite set of atomic propositions. The (time-dependent) *labeling function* $L_t : \mathcal{X} \to 2^{AP}$ maps the continuous part of each state to the set of atomic propositions that are *True* at time $t$. Each atomic proposition $\psi \in AP$ is represented by a union of polyhedrons. The finite index set $I_t^\psi$ lists the polyhedrons where $\psi$ holds at time $t$. The $i$-th polyhedron is $\{x \in \mathcal{X} \mid H_t^{\psi_i} x \le K_t^{\psi_i}\}$, where $i \in I_t^\psi$. Thus, the set of states where atomic proposition $\psi$ holds at time $t$ is given by $[\![\psi]\!](t) := \{x \in \mathcal{X} \mid H_t^{\psi_i} x \le K_t^{\psi_i}$ for some $i \in I_t^\psi\}$. This (potentially) time-varying set is the finite union of polyhedrons (finite conjunctions of halfspaces).

### 2.2 A fragment of temporal logic

We do not attempt to reason about all possible temporal logic formulas (see [3]); instead, we develop a useful library of temporal operators for robotic tasks. This fragment of temporal logic can concisely and unambiguously specify a wide range of tasks such as safe navigation, surveillance, persistent coverage, response to the environment, and visiting goals. In the following definitions, $\psi$, $\phi$, and $\psi_j$ (for a finite number of indices $j$) are propositional formulas. To simplify the presentation, we split these into three groups: core $\Phi_{\text{core}}$, response $\Phi_{\text{resp}}$, and fairness $\Phi_{\text{fair}}$. We first define the syntax of the temporal operators and then their semantics.

**Syntax:**

The core operators, $\Phi_{\text{core}} := \{\varphi_{\text{safe}}, \varphi_{\text{goal}}, \varphi_{\text{per}}, \varphi_{\text{live}}, \varphi_{\text{until}}\}$, specify fundamental properties such as safety, guarantee, persistence, liveness (recurrence), and until. These operators are,

$$\varphi_{\text{safe}} := \Box\psi, \quad \varphi_{\text{goal}} := \Diamond\psi, \quad \varphi_{\text{per}} := \Diamond\Box\psi, \quad \varphi_{\text{live}} := \Box\Diamond\psi, \quad \varphi_{\text{until}} := \psi\,\mathcal{U}\,\phi,$$

where $\varphi_{\text{safe}}$ specifies safety, i.e., a property should invariantly hold, $\varphi_{\text{goal}}$ specifies goal visitation, i.e., a property should eventually hold, $\varphi_{\text{per}}$ specifies persistence, i.e., a property should eventually hold invariantly, and $\varphi_{\text{live}}$ specifies liveness (recurrence), i.e., a property should hold repeatedly, as in surveillance, and $\varphi_{\text{until}}$ specifies until, i.e., a property $\psi$ should hold until another property $\phi$ holds.

The response operators, $\Phi_{\text{resp}} := \{\varphi^1_{\text{resp}}, \varphi^2_{\text{resp}}, \varphi^3_{\text{resp}}, \varphi^4_{\text{resp}}\}$, specify how the system responds to the environment. These operators are,

$$\varphi^1_{\text{resp}} := \Box(\psi \implies \bigcirc\phi), \qquad \varphi^2_{\text{resp}} := \Box(\psi \implies \Diamond\phi),$$
$$\varphi^3_{\text{resp}} := \Diamond\Box(\psi \implies \bigcirc\phi), \qquad \varphi^4_{\text{resp}} := \Diamond\Box(\psi \implies \Diamond\phi),$$

where $\varphi^1_{\text{resp}}$ specifies next-step response to the environment, $\varphi^2_{\text{resp}}$ specifies eventual response to the environment, $\varphi^3_{\text{resp}}$ specifies steady-state next-step response to the environment, and $\varphi^4_{\text{resp}}$ specifies steady-state eventual response to the environment.

Finally, the fairness operators, $\Phi_{\text{fair}} := \{\varphi^1_{\text{fair}}, \varphi^2_{\text{fair}}, \varphi^3_{\text{fair}}\}$, allow one to specify conditional tasks. These operators are,

$$\varphi^1_{\text{fair}} := \Diamond\psi \implies \bigwedge_{j=1}^{m} \Diamond\phi_j, \qquad \varphi^2_{\text{fair}} := \Diamond\psi \implies \bigwedge_{j=1}^{m} \Box\Diamond\phi_j,$$
$$\varphi^3_{\text{fair}} := \Box\Diamond\psi \implies \bigwedge_{j=1}^{m} \Box\Diamond\phi_j,$$

where $\varphi^1_{\text{fair}}$ specifies conditional goal visitation, and $\varphi^2_{\text{fair}}$ and $\varphi^3_{\text{fair}}$ specify conditional repeated goal visitation.

The fragment of LTL that we consider is built from the temporal operators defined above as follows,

$$\varphi ::= \varphi_{\text{core}} \mid \varphi_{\text{resp}} \mid \varphi_{\text{fair}} \mid \varphi_1 \wedge \varphi_2, \tag{2}$$

where $\varphi_{\text{core}} \in \Phi_{\text{core}}$, $\varphi_{\text{resp}} \in \Phi_{\text{resp}}$, and $\varphi_{\text{fair}} \in \Phi_{\text{fair}}$.

This LTL fragment specifies many properties relevant to robotics, especially for surveillance tasks for which no mathematical programming-based approaches currently exist. However, it does not include nested properties [3]. Determining all temporal properties that can be expressed in this framework is future work.

*Remark 1.* To include disjunctions (e.g., $\varphi_1 \vee \varphi_2$), one can rewrite a formula in disjunctive normal form, where each clause is of the form (2). In what follows, each clause can then be considered separately, as the system (1) is deterministic.

**Semantics:**

We use set operations between a trajectory (run) $x = x(x_0, u)$ and subsets of $\mathcal{X}$ where particular propositional formulas hold to define satisfaction of a temporal logic formula [3]. We denote the set of states where propositional formula $\psi$ holds by $[\![\psi]\!]$. A run $x$ *satisfies* the temporal logic formula $\varphi$, denoted by $x \vDash \varphi$, if and only if certain set operations hold. Given propositional formulas $\psi$ and $\phi$, we relate satisfaction of (a partial list of) formulas of the form (2) with set operations as follows:

- $x \vDash \square\psi$ iff $x_i \in [\![\psi]\!]$ for all $i$,
- $x \vDash \diamond\square\psi$ iff there exists an index $j$ such that $x_i \in [\![\psi]\!]$ for all $i \geq j$,
- $x \vDash \diamond\psi$ iff $x_i \in [\![\psi]\!]$ for some $i$,
- $x \vDash \square\diamond\psi$ iff $x_i \in [\![\psi]\!]$ for infinitely many $i$,
- $x \vDash \psi\,\mathcal{U}\,\phi$ iff there exists an index $j$ such that $x_j \in [\![\phi]\!]$ and $x_i \in [\![\psi]\!]$ for all $i < j$,
- $x \vDash \square(\psi \implies \bigcirc\phi)$ iff $x_i \notin [\![\psi]\!]$ or $x_{i+1} \in [\![\phi]\!]$ for all $i$,
- $x \vDash \square(\psi \implies \diamond\phi)$ iff $x_i \notin [\![\psi]\!]$ or $x_k \in [\![\phi]\!]$ for some $k \geq i$ for all $i$,
- $x \vDash \diamond\square(\psi \implies \bigcirc\phi)$ iff there exists an index $j$ such that $x_i \notin [\![\psi]\!]$ or $x_{i+1} \in [\![\phi]\!]$ for all $i \geq j$,
- $x \vDash \diamond\square(\psi \implies \diamond\phi)$ iff there exists an index $j$ such that $x_i \notin [\![\psi]\!]$ or $x_k \in [\![\phi]\!]$ for some $k \geq i$ for all $i \geq j$.

A run $x$ *satisfies* a conjunction of temporal logic formulas $\varphi = \bigwedge_{i=1}^{m} \varphi_i$ if and only if the set operations for each temporal logic formula $\varphi_i$ holds. The LTL formula $\varphi$ is *satisfiable* by a system at state $x_0 \in \mathcal{X}$ if and only if there exists a control input sequence $u$ such that $x(x_0, u) \vDash \varphi$.

## 3 Problem statement

In this section, we formally state both a feasibility and an optimization problem and give an overview of our solution approach. Let $\varphi$ be an LTL formula of the form (2) defined over $AP$.

**Problem 1.** Given a system of the form (1), with initial condition $x_0$, and an LTL formula $\varphi$ of the form (2), determine whether or not there exists a control input sequence $u$ such that $x(x_0, u) \vDash \varphi$.

We now introduce a cost function to distinguish among all trajectories that satisfy Problem 1. Since LTL formulas are defined over infinite state sequences, we define a cost function over infinite state sequences. We use a maximum cost function to simplify the presentation; it can easily be extended to discounted, limit-maximum, and average cost functions (see [26]). Let the cost $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be bounded.

**Definition 1.** Let $x$ be a trajectory and $u$ be the corresponding control input sequence. The *maximum cost* of trajectory $x$ is

$$J(x,u) := \sup_{t \in \mathcal{T}^\infty} c(x_t, u_t), \tag{3}$$

where $J$ maps trajectories and control inputs to $\mathbb{R} \cup \infty$.

**Problem 2.** Given a system of the form (1), with initial condition $x_0$, and an LTL formula $\varphi$ of the form (2), compute a control input sequence $u$ such that $x(x_0,u) \models \varphi$ and $J(x(x_0,u),u)$ is minimized.

We now give a brief overview of our solution approach. We parameterize the system trajectory (control input) as a periodic prefix-suffix structure. Every LTL operator of the form (2) is encoded as mixed-integer linear constraints on this finite parameterization. These temporal logic constraints (see Section 5) are then combined with dynamic constraints (see Section 6) as constraints on a combined mixed-integer optimization problem with an appropriate cost function. For MLD systems and certain differentially flat systems (see Section 6) with linear costs, Problems 1 and 2 can thus be solved using a mixed-integer linear program (MILP) solver. While even checking feasibility of a MILP is NP-hard, modern solvers using branch and bound methods routinely solve large problems [1]. We show promising results (see Figure 1) on high-dimensional (10+ continuous state) systems in Section 7.

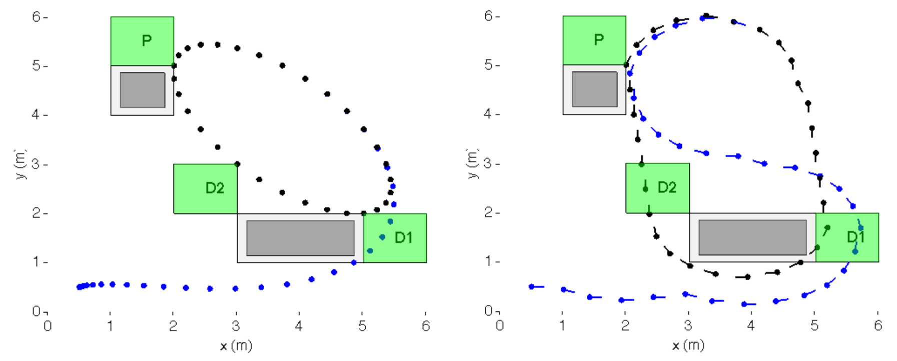

Fig. 1: Illustration of a problem instance. The task is to repeatedly visit regions $P$, $D1$, and $D2$, where dark regions are obstacles that must be avoided. Representative trajectories for a quadrotor (left) and nonlinear car (right) are shown with the prefix (blue) and suffix (black).

*Remark 2.* We only consider open-loop trajectory generation, which is already a challenging problem due to the nonlinear dynamics and LTL specifications. Disturbances can be dealt with by wrapping a feedback controller around the trajectory. Incorporating disturbances during trajectory generation is the subject of future work.

## 4 A periodic trajectory parameterization

We parameterize the system trajectory by a periodic prefix-suffix form that is commonly used in model checking for finite systems. In this structure, the prefix is a finite trajectory and the suffix is a finite trajectory that is repeated infinitely often. This gives a sufficient condition that is amenable to computation, although it may miss valid non-periodic trajectories.

A *walk* is a finite sequence of states $x = x_0x_1x_2\dots x_N$ that satisfy the constraints in (1). A *cycle* is a walk $x = x_0x_1x_2\dots x_N$ where $f(x_N,u) = x_0$ for some $u \in \mathcal{U}$. A trajectory $x$ induces a corresponding *word* (i.e., sequence of labels) $L(x) = L_0(x_0)L_1(x_1)L_2(x_2)\dots$ through the labeling function. A word is similarly defined for a walk or cycle. We now define a trajectory in *prefix-suffix* form.

**Definition 2.** Let $x_{\text{pre}}$ be a finite walk and $x_{\text{suf}}$ be a finite cycle. A trajectory $x$ is in *prefix-suffix* form if it is of the form $x = x_{\text{pre}}(x_{\text{suf}})^{\omega}$, where $\omega$ denotes infinite repetition.

We will require that the (time-varying) labeling function $L_t$ is eventually periodic.

**Assumption 1.** There exists a finite $t' \in \mathcal{T}^{\infty}$ and a $\Omega \in \mathbb{N}$ such that $L_t = L_{t+\Omega}$ for all $t \geq t' \in \mathcal{T}^{\infty}$. We further assume that $\Omega$ is minimal among all possible values.

In the sequel, we will only consider trajectories $x = x_{\text{pre}}(x_{\text{suf}})^{\omega}$ in prefix-suffix form. While both $x_{\text{pre}}$ and $x_{\text{suf}}$ are finite, the constraint that $x_{\text{suf}}$ is a cycle allows us to repeat that sequence of states forever. Repeating the same sequence of states is a sufficient condition that the word $L(x_{\text{suf}})$ (i.e., the sequence of atomic propositions) is also repeated (using Assumption 1). However, only the word matters for the feasibility of an LTL formula, not the exact sequence of states. In fact, there may exist other trajectories that produce the same word $L(x)$, but are not eventually periodic. Our approach cannot find such trajectories, although we have not noticed this limitation in our experiments. This differs from the case of finite discrete systems, where a prefix-suffix form is sufficient to find a feasible solution if one exists [3].

In the next section, we will encode the temporal operators as mixed-integer constraints on $x_{\text{pre}}$ and $x_{\text{suf}}$. Let $x_{\text{cat}} := x_{\text{pre}}x_{\text{suf}}$ denote the concatenation of $x_{\text{pre}}$ and $x_{\text{suf}}$, and assign time indices to $x_{\text{cat}}$ as $\mathcal{T}_{\text{cat}} := \{0,1,\dots,T_s,\dots,T\}$. Let $\mathcal{T}_{\text{pre}} := \{0,1,\dots,T_s - 1\}$ and $\mathcal{T}_{\text{suf}} := \{T_s,\dots,T\}$, where $T_s$ is the first time instance on the suffix. The infinite repetition of $x_{\text{suf}}$ is enforced by the constraint $x_{\text{cat}}(T_s) = f(x_{\text{cat}}(T),u)$ for some $u \in \mathcal{U}$. By Assumption 1, it is sufficient that $T_s$ is greater than $t'$ and that the length of $\mathcal{T}_{\text{suf}}$ is an integer multiple of $\Omega$. We often identify $x_{\text{pre}}(0)\cdots x_{\text{pre}}(T_{\text{pre}})$ with $x_{\text{cat}}(0)\cdots x_{\text{cat}}(T_s - 1)$ and $x_{\text{suf}}(0)\cdots x_{\text{suf}}(T_{\text{suf}})$ with $x_{\text{cat}}(T_s)\cdots x_{\text{cat}}(T)$ in the obvious manner.

## 5 A mixed-integer linear formulation of LTL constraints

In this section, we develop a mixed-integer programming formulation for a given prefix-suffix trajectory parameterization, $x_{\text{cat}} = x_{\text{pre}} x_{\text{suf}}$. The corresponding system trajectory is $x = x_{\text{pre}}(x_{\text{suf}})^{\omega}$. Since the system is deterministic, this defines a corresponding control input sequence. The split between $x_{\text{pre}}$ and $x_{\text{suf}}$ can either be specified *a priori* or left as a variable (see [26] for details). We mix notation in the following and refer to $x$ and $\mathcal{T}$ instead of $x_{\text{cat}}$ and $\mathcal{T}_{\text{cat}}$ when clear from context.

### 5.1 Relating the dynamics and propositions

We now relate the state of a system to the set of atomic propositions that are *True* at each time instance. We assume that each propositional formula $\psi$ is described at time $t$ by the union of a finite number of polytopes, indexed by the finite index set $I_t^{\psi}$. Let $[\![\psi]\!](t) := \{x \in \mathcal{X} \mid H_t^{\psi_i} x \le K_t^{\psi_i} \text{ for some } i \in I_t^{\psi}\}$ represent the set of states that satisfy propositional formula $\psi$ at time $t$. We assume that these have been constructed as necessary from the system's original atomic propositions. We note that a proposition preserving partition [2] is not necessary or even desired.

For each propositional formula $\psi$, introduce binary variables $z_t^{\psi_i} \in \{0,1\}$ for all $i \in I_t^{\psi}$ and for all $t \in \mathcal{T}$. Let $x_t$ be the state of the system at time $t$ and $M$ be a vector of sufficiently large constants. The *big-M* formulation

$$\begin{aligned} & H_t^{\psi_i} x_t \le K_t^{\psi_i} + M(1 - z_t^{\psi_i}), \quad \forall i \in I_t^{\psi} \\ & \sum_{i \in I_t^{\psi}} z_t^{\psi_i} = 1 \end{aligned} \tag{4}$$

enforces the constraint that $x_t \in [\![\psi]\!](t)$ at time $t$. Define $P_t^{\psi} := \sum_{i \in I_t^{\psi}} z_t^{\psi_i}$. If $P_t^{\psi} = 1$, then $x_t \in [\![\psi]\!](t)$. If $P_t^{\psi} = 0$, then nothing can be inferred.

The big-M formulation may give poor continuous relaxations of the binary variables, i.e., $z_t^{\psi_i} \in [0,1]$, which may lead to poor performance during optimization [1]. Such relaxations are frequently used during the solution of mixed-integer linear programs [1]. Thus, we introduce an alternate representation whose continuous relaxation is the convex hull of the original set $[\![\psi]\!](t)$. This formulation is well-known in the optimization community [13], but does not appear in the trajectory generation literature ([9, 21, 24] and references therein). As such, this formulation may be of independent interest for trajectory planning with obstacles.

The *convex hull* formulation

$$\begin{aligned} & H_t^{\psi_i} x_t^i \le K_t^{\psi_i} z_t^{\psi_i}, \quad \forall i \in I_t^{\psi} \\ & \sum_{i \in I_t^{\psi}} z_t^{\psi_i} = 1, \\ & \sum_{i \in I_t^{\psi}} x_t^i = x_t \end{aligned} \tag{5}$$

represents the same set as the big-M formulation (4). While the convex hull formulation introduces more continuous variables, it gives the tightest linear relaxation of the disjunction of the polytopes and reduces the need to select the $M$ parameters [13]. Note that we will only use the convex hull formulation (5) for safety and persistence formulas (i.e., $\varphi_{\text{safe}}$ and $\varphi_{\text{per}}$) in Section 5.2, as $P_t^{\psi} = 0$ enforces $x = 0$.

Regardless if one uses the big-M or convex hull formulation, only one binary variable is needed for each polyhedron (i.e., finite conjunction of halfspaces). This compares favorably with the approach in [16], where a binary variable is introduced for each halfspace. Additionally, the auxiliary continuous variables and mixed-integer constraints previously used are not needed because we use implication. For simple tasks such as $\varphi = \Diamond\psi$, our method can use significantly fewer binary variables than previously needed, depending on the number of halfspaces and polytopes needed to describe $[\![\psi]\!]$.

For every temporal operator described in the following section, the constraints in (4) or (5) should be understood to be implicitly applied to the corresponding propositional formulas so that $P_t^{\psi} = 1$ implies that the system satisfies $\psi$ at time $t$. Also, note that we use different binary variables for each formula—even when representing the same set.

## 5.2 The mixed-integer linear constraints

In this section, the trajectory parameterization $x$ has been *a priori* split into a prefix $x_{\text{pre}}$ and a suffix $x_{\text{suf}}$. This assumption can be relaxed, so that the size of $x_{\text{pre}}$ and $x_{\text{suf}}$ are optimization variables (see [26] for details). We further assume that $x_{\text{pre}}$ and $x_{\text{suf}}$ satisfy Assumption 1.

In the following, the correctness of the constraints applied to $x_{\text{pre}}$ and $x_{\text{suf}}$ comes directly from the temporal logic semantics given in Section 2.2 and the form of the trajectory $x = x_{\text{pre}}(x_{\text{suf}})^{\omega}$. The most important factors are whether a property can be verified over finite- or infinite-horizons. All infinite-horizon (liveness) properties must be satisfied on the suffix $x_{\text{suf}}$.

We begin with the fundamental temporal operators $\Phi_{\text{core}}$. Safety and persistence require a mixed-integer linear constraint for each time step, while guarantee and liveness only require a single mixed-integer linear constraint.

Safety, $\varphi_{\text{safe}} = \Box\psi$, is satisfied by the constraints

$$\begin{aligned} P_t^{\psi} &= 1, \quad \forall t \in \mathcal{T}_{\text{pre}}, \\ P_t^{\psi} &= 1, \quad \forall t \in \mathcal{T}_{\text{suf}}, \end{aligned}$$

which ensure that the system is always in a $[\![\psi]\!]$ region. Similarly, persistence, $\varphi_{\text{per}} = \Diamond\Box\psi$, is enforced by

$$P_t^{\psi} = 1, \quad \forall t \in \mathcal{T}_{\text{suf}},$$

which ensures the system eventually remains in a $[\![\psi]\!]$ region.

Guarantee, $\varphi_{\text{goal}} = \Diamond \psi$, is satisfied by the constraints

$$\sum_{t \in \mathcal{T}_{\text{pre}}} P_t^{\psi} + \sum_{t \in \mathcal{T}_{\text{suf}}} P_t^{\psi} = 1,$$

which ensures the system eventually visits a $[\![\psi]\!]$ region. Similarly, liveness $\varphi_{\text{live}} = \Box \Diamond \psi$ is enforced by

$$\sum_{t \in \mathcal{T}_{\text{suf}}} P_t^{\psi} = 1,$$

which ensures the system repeatedly visits a $[\![\psi]\!]$ region.

Until, $\varphi_{\text{until}} = \psi \, \mathcal{U} \, \phi$, is enforced by

$$\begin{aligned} P_0^{\phi} &= s_0, \\ P_t^{\phi} &= s_t - s_{t-1}, \quad t = 1, \ldots, T \\ P_t^{\psi} &= 1 - s_t, \quad \forall t \in \mathcal{T}, \end{aligned}$$

where we use auxiliary binary variables $s_t \in \{0,1\}$ for all $t \in \mathcal{T}$ such that $s_t \leq s_{t+1}$ for $t = 0, \ldots, T-1$ and $s_T = 1$.

Now consider the response temporal operators $\Phi_{\text{resp}}$. For these formulas, the definition of implication is used to convert each inner formula into a disjunction between a property that holds at a state and a property that holds at some point in the future. The response formulas require a mixed-integer linear constraint for each time step.

For next-step response, $\varphi^1_{\text{resp}} = \Box(\psi \implies \bigcirc \phi) = \Box(\neg \psi \vee \bigcirc \phi)$, the additional constraints are

$$\begin{aligned} P_t^{\neg\psi} + P_{t+1}^{\phi} &= 1, \quad t = 0, \ldots, T_s, \ldots, T-1, \\ P_T^{\neg\psi} + P_{T_s}^{\phi} &= 1, \end{aligned}$$

Similarly, steady-state next-step response, $\varphi^3_{\text{resp}} = \Diamond \Box (\psi \implies \bigcirc \phi) = \Diamond \Box (\neg \psi \vee \bigcirc \phi)$, is satisfied by

$$\begin{aligned} P_t^{\neg\psi} + P_{t+1}^{\phi} &= 1, \quad t = T_s, \ldots, T-1, \\ P_T^{\neg\psi} + P_{T_s}^{\phi} &= 1, \end{aligned}$$

Eventual response, $\varphi^2_{\text{resp}} = \Box(\psi \implies \Diamond \phi) = \Box(\neg \psi \vee \Diamond \phi)$, requires the following constraints

$$\begin{aligned} P_t^{\neg\psi} + \sum_{\tau = t}^{T} P_{\tau}^{\phi} &= 1, \quad \forall t \in \mathcal{T}_{\text{pre}}, \\ P_t^{\neg\psi} + \sum_{t \in \mathcal{T}_{\text{suf}}} P_t^{\phi} &= 1, \quad \forall t \in \mathcal{T}_{\text{suf}}. \end{aligned}$$

Similarly, for steady-state eventual response, $\varphi^4_{\text{resp}} = \Diamond\Box(\psi \implies \Diamond\phi) = \Diamond\Box(\neg\psi \vee \Diamond\phi)$, the additional constraints are

$$P_t^{\neg\psi} + \sum_{t\in\mathcal{T}_{\text{suf}}} P_t^{\phi} = 1, \quad \forall t \in \mathcal{T}_{\text{suf}}.$$

Now consider the fairness temporal operators $\Phi_{\text{fair}}$. In the following, the definition of implication is used to rewrite the inner formula as disjunction between a single safety (persistence) property and a conjunction of guarantee (liveness) properties. These formulas require a mixed-integer linear constraint for each conjunction in the response and each time step.

Conditional goal visitation, $\varphi^1_{\text{fair}} = \Diamond\psi \implies \bigwedge_{j=1}^m \Diamond\phi_j = \Box\neg\psi \vee \bigwedge_{j=1}^m \Diamond\phi_j$, is specified by

$$P_t^{\neg\psi} + \sum_{t\in\mathcal{T}} P_t^{\phi_j} = 1, \quad \forall j = 1,\ldots,m, \forall t \in \mathcal{T}.$$

Conditional repeated goal visitation, $\varphi^2_{\text{fair}} = \Diamond\psi \implies \bigwedge_{j=1}^m \Box\Diamond\phi_j = \Box\neg\psi \vee \bigwedge_{j=1}^m \Box\Diamond\phi_j$, is enforced as

$$P_t^{\neg\psi} + \sum_{t\in\mathcal{T}_{\text{suf}}} P_t^{\phi_j} = 1, \quad \forall j = 1,\ldots,m, \forall t \in \mathcal{T}.$$

Similarly, $\varphi^3_{\text{fair}} = \Box\Diamond\psi \implies \bigwedge_{j=1}^m \Box\Diamond\phi_j = \Diamond\Box\neg\psi \vee \bigwedge_{j=1}^m \Box\Diamond\phi_j$, is represented by

$$P_t^{\neg\psi} + \sum_{t\in\mathcal{T}_{\text{suf}}} P_t^{\phi_j} = 1, \quad \forall j = 1,\ldots,m, \ \forall t \in \mathcal{T}_{\text{suf}}.$$

We have encoded the temporal logic specifications on the system variables using mixed-integer linear constraints. Note that the equality constraints on the binary variables dramatically reduce search space. In Section 6 we discuss adding dynamics to further constrain the possible behaviors of the system.

## 6 System dynamics

The mixed-integer constraints in Section 5 are over a sequence of states, and thus are independent of the specific system dynamics. Dynamic constraints on the sequence of states can also be enforced by standard transcription methods [6]. However, the resulting optimization problem may then be a mixed-integer *nonlinear* program due to the dynamics. We highlight two useful classes of nonlinear systems where the dynamics can be encoded using mixed-integer *linear* constraints.

### 6.1 Mixed Logical Dynamical systems

Mixed Logical Dynamical (MLD) systems have both continuous and discrete-valued states and allow one to model nonlinearities, logic, and constraints [5]. These systems include constrained linear systems, linear hybrid automata, and piecewise affine systems. An MLD system is of the form

$$\begin{aligned} x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \\ \text{subject to} \quad E_2\delta(t) + E_3z(t) &\leq E_1u(t) + E_4x(t) + E_5, \end{aligned}$$

where $t \in \mathcal{T}^\infty$, $x \in \mathcal{X} \subseteq \mathbb{R}^{n_c} \times \{0,1\}^{n_l}$ are the continuous and binary states, $u \in \mathcal{U} \subseteq \mathbb{R}^{m_c} \times \{0,1\}^{m_l}$ are the inputs, and $\delta \in \{0,1\}^{r_l}$ and $z \in \mathbb{R}^{r_l}$ are auxiliary binary and continuous variables, respectively. The terms $A$, $B_1$, $B_2$, $B_3$, $E_1$, $E_2$, $E_3$, $E_4$, and $E_5$ are system matrices of appropriate dimension. We assume that the system is deterministic and well-posed (see Definition 1 in [5]).

### 6.2 Differentially flat systems

A system is *differentially flat* if there exists a set of outputs such that all states and control inputs can be determined from these outputs without integration. If a system has states $x \in \mathbb{R}^n$ and control inputs $u \in \mathbb{R}^m$, then it is flat if we can find outputs $y \in \mathbb{R}^m$ of the form $y = y(x, u, \dot{u}, \ldots, u^{(p)})$ such that $x = x(y, \dot{y}, \ldots, y^{(q)})$ and $u = u(y, \dot{y}, \ldots, y^{(q)})$. Thus, we can plan trajectories in output space and then map these to control inputs [18].

Differentially flat systems may be encoded using mixed integer linear constraints in certain cases, e.g., the flat output is constrained by mixed integer linear constraints. This holds for relevant classes of robotic systems, including quadrotors and car-like robots. However, control input constraints are typically non-convex in the flat output. Common approaches to satisfy control constraints are to plan a sufficiently smooth trajectory or slow down along a trajectory [20].

## 7 Examples

We demonstrate our techniques on a variety of motion planning problems. The first example is a chain of integrators parameterized by dimension. Our second example is a quadrotor model from [25]. Our final example is a nonlinear car-like vehicle with drift. All computations were done on a laptop with a 2.4 GHz dual-core processor and 4 GB of memory using CPLEX [1] through Yalmip [19].

The environment and task is motivated by a pickup and delivery scenario. All properties should be understood to be with respect to regions in the plane (see Figure 1). Let $P$ be a region where supplies can be picked up and $D1$ and $D2$ be regions where supplies must be delivered. The robot must remain in the safe region $S$ (in white). Formally, the task specification is $\varphi = \Box S \wedge \Box\Diamond P \wedge \Box\Diamond D1 \wedge \Box\Diamond D2$. Additionally, we minimize the maximum cost function (3) where $c(x_t, u_t) = |u_t|$ penalizes the control input.

In the remainder of this section, we consider this temporal logic motion planning problem for different system models. We use the simultaneous (sim.) approach described in Section 5.2, and also a sequential (seq.) approach from [26] that first computes the suffix and then the prefix. A trajectory of length 60 (split evenly between the prefix and suffix) is used in all cases, and all results are averaged over 20 randomly generated environments. The simultaneous approach uses between 300 and 469 binary variables with a mean of 394. Finally, all continuous-time models are discretized using a first-order hold and time-step of 0.5 seconds.

## 7.1 Chain of integrators

The first system is a chain of orthogonal integrators in the $x$ and $y$ directions. The $k$-th derivative of the $x$ and $y$ positions are controlled, i.e., $x^{(k)} = u_x$ and $y^{(k)} = u_y$, subject to the constraints $|u_x| \leq 0.5$ and $|u_y| \leq 0.5$. The general state constraints are $|x^{(i)}| \leq 1$ and $|y^{(i)}| \leq 1$ for $i = 1, \ldots, k-1$. Results are given in Tables 1 and 2 under "chain-2," "chain-6," and "chain-10," where "chain-$k$" indicates that the $k$-th derivative in both the $x$ and $y$ positions is controlled.

| | | Feasible soln. (sec) | | Num. solved | |
|---|---|---|---|---|---|
| Model | Dim. | Sim. | Seq. | Sim. | Seq. |
| chain-2 | 4 | 1.10 ±.09 | 0.64 ±.06 | 20 | 20 |
| chain-6 | 12 | 4.70 ±.48 | 2.23 ±.15 | 20 | 20 |
| chain-10 | 20 | 9.38 ±1.6 | 3.74 ±.29 | 20 | 19 |
| quadrotor | 10 | 4.20 ±.66 | 1.80 ±.15 | 20 | 20 |
| quadrotor-flat | 10 | 2.26 ±.36 | 1.99 ±1.0 | 20 | 20 |
| car-3 | 3 | 43.9 ±.77 | 10.7 ±2.0 | 4 | 20 |
| car-4 | 3 | 42.4 ±1.7 | 18.7 ±3.1 | 2 | 18 |
| car-flat | 3 | 15.8 ±3.8 | 14.0 ±4.4 | 12 | 14 |

Table 1: Time until a feasible solution was found (mean ± standard error) and number of problems (out of 20) solved in 45 seconds using the big-M formulation (4) with M = 10.

## 7.2 Quadrotor

We now consider the quadrotor model used in [25] for point-to-point motion planning, to which we refer the reader for a complete description of the model. The state $x = (p, v, r, w)$ is 10-dimensional, consisting of position $p \in \mathbb{R}^3$, velocity $v \in \mathbb{R}^3$, orientation $r \in \mathbb{R}^2$, and angular velocity $w \in \mathbb{R}^2$. This model is the linearization of a nonlinear model about hover with the yaw constrained to be zero. The control input

| | | Feasible soln. (sec) | | Num. solved | |
|---|---|---|---|---|---|
| Model | Dim. | Sim. | Seq. | Sim. | Seq. |
| chain-2 | 4 | 1.94 ±.23 | 0.94 ±.11 | 20 | 20 |
| chain-6 | 12 | 12.4 ±2.7 | 2.89 ±.32 | 20 | 20 |
| chain-10 | 20 | 16.9 ±3.0 | 7.28 ±1.2 | 17 | 15 |
| quadrotor | 10 | 18.9 ±3.8 | 2.80 ±.35 | 16 | 20 |
| car-3 | 3 | 37.3 ±3.1 | 13.3 ±1.6 | 8 | 20 |

Table 2: Time until a feasible solution was found (mean ± standard error) and number of problems (out of 20) solved in 45 seconds using the convex hull formulation (5).

$u \in \mathbb{R}^3$ is the total, roll, and pitch thrust. Results are given in Tables 1 and 2 under "quadrotor," and a sample trajectory is shown in Figure 1.

Also, we use the fact that the quadrotor is differentially flat [20] to generate trajectories for the nonlinear model (with fixed yaw). We parameterize the flat output $p \in \mathbb{R}^3$ with eight piecewise polynomials of degree three, and then optimize over their coefficients to compute a smooth trajectory. Afterwards, we check that the trajectory does not violate the control input constraints. Results are given in Table 1 under "quadrotor-flat."

## 7.3 Nonlinear car

Consider a nonlinear car-like vehicle with state $x = (p_x, p_y, \theta)$ and dynamics $\dot{x} = (v\cos(\theta), v\sin(\theta), u)$. The variables $p_x, p_y$ are position (m) and $\theta$ is orientation (rad). The vehicle's speed $v$ is fixed at 0.8 (m/s) and its control input is constrained as $|u| \leq 2.5$. We form a hybrid MLD model by linearizing the system about different orientations $\hat{\theta}_i$ for $i = 1, \ldots, k$. The dynamics are governed by the closest linearization to the current $\theta$. Results with $k = 3$ and $k = 4$ are are given in Table 1 under "car-3" and "car-4," respectively. A sample trajectory of "car-4" is show in Figure 1.

Additionally, we use the flat output $(x, y) \in \mathbb{R}^2$ to generate trajectories for the nonlinear car-like model in a similar manner as for the quadrotor model. Results are given in Table 1 under "car-flat."

## 7.4 Discussion and comparison

We first compare our approach to reachability-based algorithms that compute a finite abstraction [17, 29]. We used the method in [29] to compute a discrete abstraction for a two dimensional system in 22 seconds, and [17] reports abstracting a four dimensional system in just over a minute. This contrasts with our mixed-integer approach that can routinely find solutions to such problems in seconds, although we do not compute a feedback controller. Our results appear particularly promising for situations where the environment is dynamically changing and a finite abstraction must be repeatedly computed.

We also compare to the finite-horizon mixed-integer formulation given in [16]. Consider the task $\varphi = \Diamond \psi$, where $[\![ \psi ]\!]$ is a convex polytope defined by $m$ halfspaces. Our method uses one binary variable at each time step, while their approach uses $m$. Additionally, while we encode eventually ($\Diamond$) using a single constraint, their approach uses a number of constraints quadratic in the the trajectory length.

In most of our examples, we are able to quickly compute a feasible trajectory that satisfies a temporal logic formula by solving a mixed-integer linear program. This is aided by the sequential approach, which separates the problem into computing a suffix and then a prefix [26]. It typically takes a long time to compute a trajectory that is provably globally optimal, although this does happen in finite time.

Finally, the convex hull formulation performed poorly in our examples. There is an empirical tradeoff between having tighter continuous relaxations and the number of continuous variables in the formulation. We hypothesize that the convex hull formulation will be most useful in cases when 1) the number of binary variables is large, or 2) the cost function is minimized near the boundary of the region.

## 8 Conclusion

We presented a novel mixed-integer programming-based method for control of nonlinear systems with a useful fragment of LTL that allows both finite- and infinite-horizon properties to be specified. Our method is efficient in the number of binary variables used to model the an LTL formula. Additionally, we showed the computational effectiveness of our approach on temporal logic motion planning examples.

Future work will consider reactive environments by including both continuous and discrete disturbances using a receding horizon control approach. Additionally, we will expand the space of tasks that can be specified by including additional temporal operators and timing constraints.

**Acknowledgements** The authors thank Matanya Horowitz, Scott Livingston, and Ufuk Topcu for helpful feedback. This work was supported by a NDSEG Fellowship and the Boeing Corporation.

## References

1. User's Manual for CPLEX V12.2. IBM, 2010
2. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proc. IEEE **88**(7), 971–984 (2000)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
4. Belta, C., Habets, L.C.G.J.M.: Controlling of a class of nonlinear systems on rectangles. IEEE Trans. on Automatic Control **51**, 1749–1759 (2006)
5. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. Automatica **35**, 407–427 (1999)

6. Betts, J.T.: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, 2nd edition. SIAM (2000)
7. Bhatia, A., Maly, M.R., Kavraki, L.E., Vardi, M.Y.: Motion planning with complex goals. IEEE Robotics and Automation Magazine **18**, 55–64 (2011)
8. Blair, C.E., Jeroslow, R.G., Lowe, J.K.: Some results and experiments in programming techniques for propositional logic. Computers and operations research **13**, 633–645 (1986)
9. Earl, M.G., D'Andrea, R.: Iterative MILP methods for vehicle-control problems. IEEE Transactions on Robotics **21**, 1158–1167 (2005)
10. Fainekos, G.E., Girard, A., Kress-Gazit, H., Pappas, G.J.: Temporal logic motion planning for dynamic robots. Automatica **45**, 343–352 (2009)
11. Habets, L., Collins, P.J., van Schuppen, J.H.: Reachability and control synthesis for piecewise-affine hybrid systems on simplices. IEEE Trans. on Automatic Control **51**, 938–948 (2006)
12. Hooker, J.N., Fedjki, C.: Branch-and-cut solution of inference problems in propositional logic. Annals of Mathematics and Artificial Intelligence **1**, 123–139 (1990)
13. Jeroslow, R.G.: Representability in mixed integer programming, I: Characterization results. Discrete Applied Mathematics **17**, 223–243 (1987)
14. Karaman, S., Frazzoli, E.: Linear temporal logic vehicle routing with applications to multi-UAV mission planning. Int. J. of Robust and Nonlinear Control **21**, 1372–1395 (2011)
15. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning with deterministic $\mu$-calculus specifications. In: Proc. of American Control Conf. (2012)
16. Karaman, S., Sanfelice, R.G., Frazzoli, E.: Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In: Proc. of IEEE Conf. on Decision and Control, pp. 2117–2122 (2008)
17. Kloetzer, M., Belta, C.: A fully automated framework for control of linear systems from temporal logic specifications. IEEE Trans. on Automatic Control **53**(1), 287–297 (2008)
18. LaValle, S.M.: Planning Algorithms. Cambridge Univ. Press (2006)
19. Löfberg, J.: YALMIP : A toolbox for modeling and optimization in MATLAB. In: Proc. of the CACSD Conference. Taipei, Taiwan (2004). Software available at http://control.ee.ethz.ch/~joloef/yalmip.php
20. Mellinger, D., Kushleyev, A., Kumar, V.: Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In: Proc. of Int. Conf. on Robotics and Automation (2012)
21. Richards, A., How, J.P.: Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: American Control Conference (2002)
22. Smith, S.L., Tumova, J., Belta, C., Rus, D.: Optimal path planning for surveillance with temporal-logic constraints. Int. J. of Robotics Research **30**, 1695–1708 (2011)
23. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. Philadelphia, PA: SIAM (2001)
24. Vitus, M.P., Pradeep, V., Hoffmann, J., Waslander, S.L., Tomlin, C.J.: Tunnel-MILP: path planning with sequential convex polytopes. In: Proc. of AIAA Guidance, Navigation, and Control Conference (2008)
25. Webb, D.J., van den Berg, J.: Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In: Proc. of IEEE Int. Conf. on Robotics and Automation (2013)
26. Wolff, E.M., Murray, R.M.: Optimal control of mixed logical dynamical systems with long-term temporal logic specifications. Tech. rep., California Institute of Technology (2013). URL `http://resolver.caltech.edu/CaltechCDSTR:2013.001`
27. Wolff, E.M., Topcu, U., Murray, R.M.: Optimal control with weighted average costs and temporal logic specifications. In: Proc. of Robotics: Science and Systems (2012)
28. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon temporal logic planning. IEEE Trans. on Automatic Control (2012)
29. Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., Murray, R.M.: TuLiP: A software toolbox for receding horizon temporal logic planning. In: Proc. of Int. Conf. on Hybrid Systems: Computation and Control (2011). Http://tulip-control.sf.net