Reactive Synthesis from Signal Temporal Logic Specifications

Vasumathi Raman
California Institute of Technology
Pasadena, CA, USA
vasu@caltech.edu

Alexandre Donzé
UC Berkeley
Berkeley, CA, USA
donze@berkeley.edu

Dorsa Sadigh
UC Berkeley
Berkeley, CA, USA
dsadigh@berkeley.edu

Richard M. Murray
California Institute of Technology
Pasadena, CA, USA
murray@cds.caltech.edu

Sanjit A. Seshia
UC Berkeley
Berkeley, CA, USA
sseshia@eecs.berkeley.edu

ABSTRACT

We present a counterexample-guided inductive synthesis approach to controller synthesis for cyber-physical systems subject to signal temporal logic (STL) specifications, operating in potentially adversarial nondeterministic environments. We encode STL specifications as mixed integer-linear constraints on the variables of a discrete-time model of the system and environment dynamics, and solve a series of optimization problems to yield a satisfying control sequence. We demonstrate how the scheme can be used in a receding horizon fashion to fulfill properties over unbounded horizons, and present experimental results for reactive controller synthesis for case studies in building climate control and autonomous driving.

1. INTRODUCTION

We are concerned with controlling hybrid systems to satisfy desired properties despite a potentially adversarial environment; the provided solution must be robust to environment actions with regards to which we are uncertain. Recently, temporal logics have proven a valuable tool for controller synthesis, because they provide a compact mathematical formalism for specifying desired behaviors of a system. There is a rich body of literature containing algorithms for verification and synthesis of systems obeying temporal logic specifications. Approaches can be broadly categorized based on whether they utilize a discrete abstraction of the system, and whether the environment is assumed to be deterministic.

Approaches that utilize a discrete abstraction enable construction of discrete supervisory controllers, which have successfully been used to construct hybrid controllers for domains including robotics and aircraft power system design; these include approaches that deal with deterministic [14, 20] as well as adversarial environments [7, 24]. In contrast, approaches that eschew discrete abstractions include those based on sampling-based methods [12], and mixed-integer linear programming encodings of temporal logic specifications[13, 11, 15, 23, 21]. The latter have thus far been confined to the realm of deterministic operating environments, and it is this gap that we close with the current work.

We adopt a Counterexample-Guided Inductive Synthesis [22] approach to synthesizing a controller satisfying reactive specifications. Inductive synthesis refers to the automated generation of a system from input-output examples, using each new example to iteratively refine the hypothesis about the system until convergence. In Counterexample-Guided Inductive Synthesis (CEGIS), the examples are mostly counterexamples discovered while trying to verify correctness of the current guess. CEGIS thus relies primarily on a validation engine to validate candidates produced at intermediate iterations, which can produce counterexamples for use in the next iteration. Automated synthesis of systems using CEGIS and the closely related counterexample -guided abstraction refinement (CEGAR) paradigm has been widely studied in various contexts [4, 1, 10].

The specification language adopted here is Signal Temporal Logic (STL) [18], which allows the specification of temporal properties of real-valued signals, and has been applied to the analysis of hybrid dynamical systems from various application domains such as analog and mixed signal circuits, systems biology or Cyber-Physical Systems (CPS). STL has the advantage of naturally admitting a *quantitative* semantics which, in addition to the binary answer to the question of satisfaction, provides a real number indicating the quality of the satisfaction or violation. Such quantitative semantics have been defined for timed logics like Metric Temporal Logic (MTL) [8] and STL [6] to assess the *robustness* of the systems to parameter or timing variations. We exploit this ability to compute the robustness of satisfaction in the validation engine for our CEGIS approach to reactive synthesis.

A key advantage of temporal logic over, e.g. domain-specific languages based on propositional logic, is that it allows the expression of properties of infinite traces. We would therefore like to synthesize controllers that can run indefinitely, and satisfy infinite-horizon properties. Receding Horizon Control (Receding Horizon Control (RHC)) [19] is based on iterative, finite horizon, discrete time optimization of a model of the plant: at time t, the current plant state is observed, and an optimal control strategy computed for a finite time horizon in the future, $[t, t+H]$. The first step of the computed strategy is implemented, the plant state is then sampled again, and new calculations performed on a horizon of size H starting from the new current state. This not only reduces computational complexity, but improves robustness with respect to exogenous disturbances and modeling uncertainties by allowing new information to be incorporated as it becomes available [19].

We have already made the connection between RHC and control synthesis from STL specifications in previous work[21], where we specify desired properties of the system using a STL formula, and synthesize control such that the system satisfies that specification, while using a receding horizon approach. We presented automatically-generated Mixed Integer Linear Program (MILP) encodings for STL specifications, extending the Bounded Model Checking (BMC) paradigm for finite discrete systems [3] to STL. These encodings can be used not only to generate open-loop control signals that satisfy finite and infinite horizon STL properties, but also to generate signals that maximize quantitative (robust) satisfaction. We now show how the robustness-based encoding can be used to produce a validation engine that synthesizes counterexamples to guide a CEGIS approach to reactive synthesis.

Receding horizon control to satisfy temporal logic specifications in adversarial settings has been considered before in the context of Linear Temporal Logic (LTL) [24], where the authors propose a scheme that makes use of discrete abstractions to synthesize supervisory controllers for specifications with GR(1) goals. In that work, feasibility of the global specification is determined via symbolic checks on a series of pre-defined smaller problems, and strategies extracted as needed. In contrast, we do not require an *a priori* defined finite set of sub-problems. Our approach also extends synthesis capabilities to a wider class of temporal logic specifications and environments than [9, 2], and avoids potentially expensive computations of a finite state abstraction of the system as in [5] and [24].

Contributions: We present a CEGIS approach to controller synthesis for cyber-physical systems subject to signal temporal logic (STL) specifications, operating in potentially adversarial nondeterministic environments.

- We encode STL specifications as mixed integer-linear constraints on the variables of a discrete-time model of the system and environment dynamics, and solve a counterexample guided series of optimization problems to yield a satisfying control sequence.
- Our scheme can be used in a receding horizon fashion to fulfill properties over unbounded horizons.
- We present experimental results using a case study of controller synthesis on a model of a Heating Ventilation and Air Conditioning (HVAC) system with nondeterministic elements in the environment, and an autonomous driving scenario in the presence of adversarial agents; simulation results in these two domains illustrate the effectiveness of our methodology.

2. PRELIMINARIES

We consider discrete-time continuous systems of the form

$$x_{t+1} = f(x_t, u_t, w_t) \tag{1}$$

where $t = 0, 1, \ldots$ are the time indices, $x \in \mathcal{X} \subseteq (\mathbb{R}^{n_c} \times \{0,1\}^{n_l})$ are the continuous and binary/logical states, $u \in U \subseteq (\mathbb{R}^{m_c} \times \{0,1\}^{m_l})$ are the (continuous and logical) control inputs, $w \in W \subseteq (\mathbb{R}^{e_c} \times \{0,1\}^{e_l})$ are the (possibly adversarial) external inputs or disturbances, and $x_0 \in \mathcal{X}$ is the initial state. We will refer to w as the *environment* input.

A *run* $\mathbf{f} = (x_0u_0w_0)(x_1u_1w_1)(x_2u_2w_2)...$ is an infinite sequence where $x_t \in \mathcal{X}$ is the state of the system at index t, and for each $t \in \mathbb{N}$, $u_t \in U$, $w_t \in W$ and $x_{t+1} = f(x_t, u_t, w_t)$. Given $x_0 \in X$, $\mathbf{u} \in U^\omega$ and $\mathbf{w} \in W^\omega$, denote by $\mathbf{f}(x_0, \mathbf{u}, \mathbf{w})$ the run generated following (1). The corresponding sequence of states, which we also call *signal* in the rest of the paper, is denoted by $\mathbf{x} = x_0x_1\ldots$. We assume that given an initial state $x_0 \in X$, a control input sequence $\mathbf{u}^N = u_0u_1u_2\ldots u_{N-1} \in U^N$ and a sequence of environment inputs $\mathbf{w}^N = w_0w_1w_2\ldots w_{N-1} \in W^N$, the resulting horizon-$N$ run of a system modeled by (1), which we denote by $\mathbf{f}(x_0, \mathbf{u}^N, \mathbf{w}^N) = (x_0u_0w_0)(x_1u_1w_1)(x_2u_2w_2)...(x_Nu_Nw_N)$, is unique. In addition, we introduce a generic cost function $J(\mathbf{f}(x_0, \mathbf{u}, \mathbf{w}))$ that maps (infinite and finite) runs to $\mathbb{R}$.

2.1 Signal Temporal Logic

We consider STL formulas defined recursively according to the grammar

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Box_{[a,b]}\ \psi \mid \varphi\ \mathcal{U}_{[a,b]}\ \psi$$

where μ is a predicate whose value is determined by the sign of a function of an underlying signal $\mathbf{x}$, i.e., $\mu \equiv \mu(\mathbf{x}) > 0$, and ψ is an STL formula. The validity of a formula φ with respect to signal $\mathbf{x}$ at time t is defined inductively as follows:

$$\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & \mu(x_t) > 0 \\
(\mathbf{x}, t) \models \neg\mu & \Leftrightarrow & \neg((\mathbf{x}, t) \models \mu) \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (\mathbf{x}, t) \models \varphi \wedge (\mathbf{x}, t) \models \psi \\
(\mathbf{x}, t) \models \varphi \vee \psi & \Leftrightarrow & (\mathbf{x}, t) \models \varphi \vee (\mathbf{x}, t) \models \psi \\
(\mathbf{x}, t) \models \Box_{[a,b]}\, \varphi & \Leftrightarrow & \forall t' \in [t+a, t+b], (\mathbf{x}, t') \models \varphi \\
(\mathbf{x}, t) \models \varphi\ \mathcal{U}_{[a,b]}\ \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ s.t. } (\mathbf{x}, t') \models \psi \\
 & & \wedge \forall t'' \in [t, t'], (\mathbf{x}, t'') \models \varphi.
\end{array}$$

A signal $\mathbf{x} = x_0x_1x_2...$ satisfies φ, denoted by $\mathbf{x} \models \varphi$, if $(\mathbf{x}, 0) \models \varphi$. Informally, $\mathbf{x} \models \Box_{[a,b]}\, \varphi$ if φ holds at every time step between a and b, and $\mathbf{x} \models \varphi\ \mathcal{U}_{[a,b]}\ \psi$ if φ holds at every time step before ψ holds, and ψ holds at some time step between a and b. Additionally, we define $\Diamond_{[a,b]}\, \varphi = \top\ \mathcal{U}_{[a,b]}\ \varphi$, so that $\mathbf{x} \models \Diamond_{[a,b]}\, \varphi$ if φ holds at some time step between a and b. Note that since we deal only with discrete-time systems, the STL formulas we consider refer only to intervals over discrete time values. However one of the advantages of the STL and its accompanying quantitative semantics is that it enables assessment of robustness to imprecisions resulting from the discretization of time [6].

An STL formula φ is *bounded-time* if it contains no unbounded operators; the *bound* of φ is the maximum over the sums of all nested upper bounds on the temporal operators, and provides a conservative maximum trajectory length required to decide its satisfiability. For example, for $\Box_{[0,10]} \Diamond_{[1,6]} \varphi$, a trajectory of length $N \geq 10+6 = 16$ is sufficient to determine whether the formula is satisfiable. This bound can be computed in time linear in the formula length.

2.2 Robust Satisfaction of STL formulas

Quantitative or robust semantics define a real-valued function ρ^φ of signal $\mathbf{x}$ and t such that $(\mathbf{x}, t) \models \varphi \equiv \rho^\varphi(\mathbf{x}, t) > 0$. This is computed recursively from the above semantics in a straightforward manner, by propagating the values of the functions associated with each operand using min and max operators corresponding to various STL operators. For example, the robust satisfaction of $\mu_1 \equiv x - 3 > 0$ at time 0 is $\rho^{\mu_1}(x, 0) = x_0 - 3$. The robust satisfaction of $\mu_1 \wedge \mu_2$ is the minimum of ρ^{μ_1} and ρ^{μ_2}. Temporal operators can be treated as conjunctions and disjunctions along the time axis, e.g., the robust satisfaction of $\varphi = \Box_{[0,2]} \mu_1$ is $\rho^\varphi(x, t) = \min_{t \in [0,2]} \rho^{\mu_1}(x, t) = \min_{t \in [0,2]} x_t - 3$. The complete robust semantics is defined as follows:

$$\begin{array}{lcl}
\rho^{\mu}(\mathbf{x}, t) & = & \mu(x_t) \\
\rho^{\neg\mu}(\mathbf{x}, t) & = & -\mu(x_t) \\
\rho^{\varphi \wedge \psi}(\mathbf{x}, t) & = & \min(\rho^{\varphi}(\mathbf{x}, t), \rho^{\psi}(\mathbf{x}, t)) \\
\rho^{\varphi \vee \psi}(\mathbf{x}, t) & = & \max(\rho^{\varphi}(\mathbf{x}, t), \rho^{\psi}(\mathbf{x}, t)) \\
\rho^{\Box_{[a,b]}\, \varphi}(\mathbf{x}, t) & = & \min_{t' \in [t+a, t+b]} \rho^{\varphi}(\mathbf{x}, t') \\
\rho^{\varphi\ \mathcal{U}_{[a,b]}\ \psi}(\mathbf{x}, t) & = & \max_{t' \in [t+a, t+b]} (\min(\rho^{\psi}(\mathbf{x}, t'), \\
 & & \qquad \min_{t'' \in [t, t']} \rho^{\varphi}(\mathbf{x}, t'')))
\end{array}$$

2.3 MILP Encoding for Controller Synthesis

In order to synthesize a run that satisfies an STL formula φ, we add STL constraints to an MILP formulation of the control synthesis problem as in [21]. To do so, we first represent the system trajectory as a finite sequence of states satisfying the model dynamics (1). Then we encode the formula φ with a set of MILP constraints; our encoding produces an MILP as long as the predicates μ in φ are linear or affine.

2.3.1 Constraints on system evolution

The system constraints encode valid finite (horizon-N) trajectories for a system of the form (1) – these constraints hold if and only if the trajectory $\mathbf{f}(x_0, \mathbf{u}_N, \mathbf{w}_N)$ satisfies (1) for $t = 0, 1, ..., N$.

2.3.2 STL constraints

The robustness of satisfaction of the STL specification, as defined in 2.2, provides a natural objective for the MILP defined in section 2.3, either in the absence of, or as a complement to domain-specific objectives on turns of the system.

As described in Section 2.2, the robustness of an STL specification φ can be computed recursively on the structure of the formula. Moreover, since max and min operations can be expressed in an MILP formulation using additional binary variables, this does not add complexity to the encoding (although the additional variables do make it more computationally expensive in practice). Given a formula φ, we introduce a variable r_t^φ, and an associated set of MILP constraints such that $r_t^\varphi > 0$ if and only if φ holds at position t. We recursively generate the MILP constraints, such that r_0^φ determines whether a formula φ holds in the initial state. Additionally, we enforce that the value of $r_t^\varphi = \rho^\varphi(\mathbf{x}, t)$. The reader is referred to [21] for details of this encoding.

The advantage of this *robustness-based* encoding is that it allows us to maximize or minimize the value of r_0^φ, obtaining a trajectory that maximizes or minimizes the robustness of satisfaction.

The union of the STL constraints and system constraints yields an MILP, enabling us to check feasibility and finding a solution when possible using an MILP solver; for further details and examples see [21]. Given an objective function on runs of the system, we can also find an optimal trajectory that satisfies the STL specification. The robustness also provides a natural objective for this MILP, either in the absence of, or as a complement to domain-specific objectives on runs of the system.

Mixed integer-linear programs are NP-hard, and hence impractical when the dimensions of the problem grow. We present the computational costs of the above encoding in terms of the number of variables and constraints in the resulting MILP. If P is the set of predicates used in the formula and $|\varphi|$ is the length (i.e. the number of operators), then $O(N \cdot |P|) + O(N \cdot |\varphi|)$ continuous variables are introduced. In addition, $O(N)$ binary variables are introduced for every instance of a Boolean operator, i.e. $O(N \cdot |\varphi|)$ Boolean variables.

3. PROBLEM STATEMENT

We address the problem of synthesizing control inputs for a system operating in the presence of potentially adversarial, a priori uncertain external inputs or disturbances. The controllers we produce will guarantee specifications of the form $\varphi \doteq \varphi_e \Rightarrow \varphi_s$, where φ_e places assumptions on the external environment, and φ_s specifies desired guarantees on the plant behavior. In this work, φ_e refers exclusively to properties of signals $\mathbf{w} \in W^\omega$, whereas φ_s refers to properties of $\mathbf{x} \in \mathcal{X}^\omega$ and $\mathbf{u} \in U^\omega$.

We now formally state the synthesis problem for reactive controllers subject to STL specifications of the form above, and its receding horizon formulation.

Problem 1 (STL Reactive Synthesis). *Given a system of the form (1), initial state x_0, trajectory length N, STL formula φ and cost function J, compute*

$$\begin{array}{lll}
\underset{\mathbf{u}^N}{\text{argmin}} & \underset{\mathbf{w}^N \in \{\mathbf{w} \in W^N | \mathbf{w} \models \varphi_e\}}{\max} & J(\mathbf{f}(x_0, \mathbf{u}^N, \mathbf{w}^N)) \\
s.t. & \forall \mathbf{w}^N \in W^N, & \mathbf{f}(x_0, \mathbf{u}^N, \mathbf{w}^N) \models \varphi
\end{array}$$

Problem 2 (Receding Horizon Reactive Synthesis). *Given a system of the form (1), initial state x_0, STL formula*

φ and cost function J, at each time step t, compute

$$\begin{aligned} &\underset{\mathbf{u}^{H,t}}{\text{argmin}} \quad \max_{\mathbf{w}^{H,t}\in\{\mathbf{w}\in W^{H,t}|\mathbf{w}\models\varphi_e\}} \quad J(\mathbf{f}(x_t,\mathbf{u}^{H,t},\mathbf{w}^{H,t})) \\ &s.t. \quad \forall \mathbf{w}\in W^{\omega}, \quad \mathbf{f}(x_0,\mathbf{u},\mathbf{w})\models\varphi, \end{aligned}$$

where H is a finite horizon provided as a user input or selected in some other fashion, $\mathbf{u}^{H,t}$ is the horizon-H control input computed at each time step and $\mathbf{u} = u_0^{H,0}u_0^{H,1}u_0^{H,2}....$

In Sections 4 and 5, we present both a finite-trajectory solution to Problem 1, and a solution to Problem 2 for a large class of STL formulas. A key component of our solution is to use the encoding of STL specifications as MILP constraints presented in [21], in combination with MILP constraints representing the system dynamics to efficiently solve the resulting constrained optimization problem.

4. COUNTEREXAMPLE-GUIDED FINITE HORIZON SYNTHESIS

We propose a solution to Problem 1 using a counterexample guided inductive synthesis (CEGIS) procedure. We first consider bounded STL properties φ, bounded by $N \in \mathbb{N}$. Once we have this scheme for synthesizing control for finite trajectories satisfying bounded specifications, we will use a receding horizon scheme for infinite trajectories.

Algorithm 1 CEGIS Algorithm for Problem 1

```
 1: procedure CEGIS($f, x_0, N, \varphi, J$)
 2:     Let $\mathbf{w}^0 = (w_1^0, w_2^0, ...w_{N-1}^0)$, s.t. $\mathbf{w}^N \models \varphi_e$
 3:     $W_{cand} = \{\mathbf{w}^0\}$
 4:     while True do
 5:
            $\mathbf{u}^0 \leftarrow \underset{\mathbf{u}\in U^N}{\text{argmin}} \; \max_{\mathbf{w}^0\in W_{cand}}(J(\mathbf{f}(x_0,\mathbf{u},\mathbf{w}^0)))$
                 s.t. $\forall \mathbf{w}^0 \in W_{cand}, \; \mathbf{f}(x_0,\mathbf{u},\mathbf{w}^0) \models \varphi_s$,
 6:         if $\mathbf{u}^0$ == null then
 7:             Return INFEASIBLE
 8:         end if
 9:
            $\mathbf{w}^1 \leftarrow \text{argmin}_{\mathbf{w}\in W^N} \; \rho^{\varphi}(\mathbf{f}(x_0,\mathbf{u}^0,\mathbf{w}),0)$
                 s.t. $\mathbf{w}^1 \models \varphi_e$
10:         if $\rho^{\varphi}(\mathbf{f}(x_0,\mathbf{u}^0,\mathbf{w}^1)) > 0$ then
11:             Return $\mathbf{u}^0$
12:         else
13:             $W_{cand} \leftarrow W_{cand} \cup \{\mathbf{w}^1\}$
14:         end if
15:     end while
16: end procedure
```

We now describe the steps of Algorithm 1 in detail. In Step 2, we choose an initial instance $\mathbf{w}^0$ of an environment that satisfies the specification φ_e. We do so using the open-loop synthesis algorithm for bounded-time STL described in [21]. Our initial set of candidate environment inputs is a singleton, $W_{cand} = \{\mathbf{w}^0\}$ (Step 3). Then, in Step 5, we compute the optimal control input $\mathbf{u}^0$ with respect to this environment, such that the system specification φ_s is satisfied; this step also uses the solution in [21]. If the problem in Step 5 is infeasible, we know that there is a control input $\mathbf{w}^0 \in W_{cand}$ against which no control input can satisfy φ, so we can stop and return (Step 7). Otherwise, in Step 9, we find an environment $\mathbf{w}^1$ that satisfies φ_e, but also minimizes the robustness of satisfaction of φ for the control input $\mathbf{u}^0$. Essentially, this step tries to find an environment that falsifies the specification φ when the control input $\mathbf{u}^0$ is used. If the minimum robustness $\rho^{\varphi}(\mathbf{f}(x_0,\mathbf{u}^0,\mathbf{w}^1))$ thus computed is positive, this implies $\mathbf{f}(x_0,\mathbf{u}^0,\mathbf{w}) \models \varphi \; \forall \mathbf{w} \in W^N$, so we can return the control input $\mathbf{u}^0$ as our result in Step 11 . Otherwise, we have generated a counterexample to $\mathbf{u}^0$ being the desired control input, i.e. an environment $\mathbf{w}^1$ that falsifies φ when $\mathbf{u}^0$ is used. We use this counterexample to guide our inductive synthesis in Step 13, by adding it to the set of environments to be considered in the next iteration. We then resume execution of the while loop.from Step 4.

Theorem 1. *If Algorithm 1 returns $\mathbf{u}^N \in U^N$, then $\forall \mathbf{w}^N \in W^N$, $\mathbf{f}(x_0,\mathbf{u}^N,\mathbf{w}^N) \models \varphi$. If Algorithm 1 returns* `INFEASIBLE`*, then Problem 1 is infeasible.*

Note that Algorithm 1 does not fully solve Problem 1, because it does not always ensure cost-optimality of $\mathbf{u}^N$ with respect to all disturbances $\mathbf{w}^N \in W^N$ — the returned $\mathbf{u}^N$ is optimal with respect to a specific set of disturbances $W_{cand} \subseteq W^N$.

Since $|W_{cand}|$ grows by 1 at every iteration of the `while` loop, the MILP in Step 5 grows linearly with the number of iterations, since we duplicate constraints for each new counterexample. If W is finite, W_{cand} will converge, and Algorithm 1 is sound and complete. Otherwise, we execute a maximum number of iterations of the while loop before declaring the problem infeasible.

In practice, solving the problem in Step 5 becomes expensive as W_{cand} grows, in particular because the objective is now non-linear. While state-of-the-art MILP solvers like Gurobi[1] handle nonlinear objective functions efficiently, we can preserve the difficulty of the problem at each iteration by setting $W_{cand} = \{\mathbf{w}^1\}$ in Step 13 instead of growing the set of candidates. This breaks completeness even for finite sets W, since we may oscillate between two disturbances, but preserves soundness with respect to the satisfaction of φ, while allowing faster solutions at each iteration of the loop.

In the case studies described in Section 6, we find that a couple of iterations through the while loop suffices to either find a satisfying control input or render the problem infeasible.

5. RECEDING HORIZON SYNTHESIS

In this section, we will describe a solution to Problem 2 by adding STL constraints to a receding horizon control framework. At each step t of the computation, we will employ the CEGIS approach in Section 4 to find a finite trajectory of fixed horizon length H, such that the trajectory accumulated over time satisfies φ.

Note that this problem is relatively simple for bounded-time

[1] http://www.gurobi.com/

STL formulas φ, as described in [21]. Here the length of the horizon H is chosen to be at least the bound of formula φ. Then at time step 0, we synthesize control $\mathbf{u}^{H,0}$ using the formulation in Section 4, and execute only the first time step $u_0^{H,0}$; we then observe $w_0^{H,0}$ and x_1. Then at the next step, we solve for $\mathbf{u}^{H,1}$, while constraining the values of $u_0^{H,1} = u_0^{H,0}, w_0^{H,1} = w_0^{H,0}$ in the MILP, and retaining the STL constraints on the trajectory up to time H. Keeping track of the history in this manner ensures that the formula is satisfied over the length-H prefix of the trajectory, while solving for $\mathbf{u}^{H,t}$ at every time step t.

Suppose now that we have a specification $\psi = \Box\,\varphi$, where φ is a bounded-time formula with bound H. In this case, we can stitch together trajectories of length H using a receding horizon approach to produce an infinite computation that satisfies the STL formula. At each step of the receding horizon computation, we search for a finite trajectory of horizon length $2H$, keeping track of the past values and robustness constraints necessary to determine satisfaction of ψ at every time step in the trajectory.

First we define a procedure

$$\texttt{CEGIS}^*(f, x_0, N, \psi = \Box\,\varphi, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$$

that takes additional inputs $\mathbf{P} = \{P_0, P_1, ..., P_{H-1}\}$ and $\mathbf{u}_{old}^t = u_0, u_1, ..., u_{t-1}$, and is identical to Algorithm 1, except that the optimization problem in Step 5 is solved with the added constraints:

$$\rho^{\varphi}(\mathbf{f}(x_0, \mathbf{u}, \mathbf{w}^0), i) > P_i \forall i \in [0, H-1]$$
$$\mathbf{u}[i...t] = \mathbf{u}_{old}^t$$

We then define a receding horizon control procedure as in Algorithm 2. At each step, we are optimizing over a horizon of $2H$.

Algorithm 2 RHC Algorithm for Problem 2

```
1:  procedure RHC($f, x_0, \psi = \Box\,\varphi, J$)
2:      Let $M$ be a large positive constant.
3:      Let $H$ be the bound of $\varphi$.
4:      Set $P_0 = 0$ and $P_i = -M \ \forall 0 < i \leq H$.
5:      Compute $\mathbf{u}^0 = u_0^0, u_1^0, ...., u_{2H-1}^0$ as:
            $\mathbf{u}^0 \leftarrow \texttt{CEGIS}^*(f, x_0, 2H, \Box_{[0,H]}\,\varphi, J, \mathbf{P}^H, \emptyset)$
6:      for t=1; t<=H;t=t+1 do
7:          Set $\mathbf{u}_{old}^t = u_0^0, u_1^1, u_2^2, ..., u_{t-1}^{t-1}$.
8:          Set $P_i = 0$ for $0 \leq i \leq t$, $P_i = -M \ \forall t < i \leq H$.
9:          Compute $\mathbf{u}^t = u_0^t, u_1^t, ...., u_{2H-1}^t$ as:
            $\mathbf{u}^t \leftarrow \texttt{CEGIS}^*(f, x_t, 2H, G_{[0,H]}\varphi, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$
10:     end for
11:     while True do
12:         Set $\mathbf{u}_{old}^t = u_1^{t-1}, u_2^{t-1}, u_3^{t-1}, ..., u_t^{t-1}$.
13:         Set $P_i = 0$ for $0 \leq i \leq H$.
            $\mathbf{u}^t \leftarrow \texttt{CEGIS}^*(f, x_t, 2H, G_{[0,H]}\varphi, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$
14:     end while
15: end procedure
```

Algorithm 2 has two phases, a *transient* phase (Lines 4-10) and a *stationary* phase (Lines 11-14). The transient phase applies until an initial control sequence of length H has been computed, and the stationary phase follows. In the transient phase, the number of stored previous inputs ($\mathbf{u}_{old}^t$) as well as the number of time steps at which formula φ is enforced (i.e. time steps for which $P_i = 0$) grows by one at each iteration, until they both attain a maximum of H at iteration H. Every following iteration uses a window of size H for stored previous inputs, and sets all $P_i = 0$. The size-H window of previously-computed inputs advances forward one step in time at each iteration after step H. In this manner, we keep a record of the previously computed inputs required to ensure satisfaction of φ up to H time steps in the past.

THEOREM 2. *Let $\mathbf{u}^*$ be the infinite sequence of control inputs generated by setting $\mathbf{u}^*[t] = u_0^t$, where $\mathbf{u}^t = u_0^t u_1^t ... u_{2H-1}^t$ is the control input sequence of length $2H$ generated by Algorithm 2 at time t. Then $\forall \mathbf{w} \in W^\omega,\ \mathbf{f}(x_0, \mathbf{u}^*, \mathbf{w}) \models \psi$.*

PROOF. Since H is the bound of φ, the satisfaction of φ at time t is established by the control inputs $\mathbf{u}^*[t : t+H-1]$. At time $t+H$,

$$\begin{aligned} \mathbf{u}_{old}^{t+H} &= u_0^{t+H}, u_1^{t+H}, u_2^{t+H}, ..., u_{t+H-1}^{t+H} \\ &= u_1^{t+H-1}, u_2^{t+H-1}, u_3^{t+H-1}, ..., u_H^{t+H-1} \\ &= u_t^t, u_{t+1}^{t+1}, u_{t+2}^{t+2}, ..., u_{t+H-1}^{t+H-1} \\ &= \mathbf{u}^*[t : t+H-1], \end{aligned}$$

and so all the inputs required to determine satisfaction of φ at time t have been fixed. Moreover, if $\mathbf{u}^{t+H}$ is successfully computed, then by the correctness of Algorithm 1, $\mathbf{u}_{old}^{t+H}$ has the property that $\forall \mathbf{w}^H \in W^H,\ \mathbf{f}(x_t, \mathbf{u}_{old}^{t+H}, \mathbf{w}^H) \models \varphi$. Since $\mathbf{u}^*[t : t+H-1] = \mathbf{u}_{old}^{t+H}$, we see that $\forall \mathbf{w}^H \in W^H,\ \mathbf{f}(x_t, \mathbf{u}^*[t : t+h], \mathbf{w}^H) \models \varphi$.

It follows that $\forall \mathbf{w}^\omega \in W^\omega,\ \mathbf{f}(x_0, \mathbf{u}^*, \mathbf{w}) \models \varphi$. □

We have therefore shown how a control input can be synthesized for infinite sequences satisfying ψ, by repeatedly synthesizing control for sequences of length $2H$. A similar approach applies for formulas $\Diamond\,\varphi$ and $\varphi\ \mathcal{U}\ \psi$, where φ, ψ are bounded-time.

6. CASE STUDIES

We now validate our approach in simulation, for case studies in building climate control and autonomous driving.

6.1 Building Climate Control

We consider the problem of controlling building indoor climate in a commercial building equipped with a HVAC system controlled by a receding horizon control scheme. We adopt the model proposed by Maasoumy et al [17], and the receding horizon control formulation proposed by Maasoumy et al. [16], with the objective of minimizing the total energy cost (in dollar value).

As shown in Figure 1, we model a building with 4 rooms; we denote the temperature of room r_i by T_i, and that of the outside by T_5. The temperature of a room is governed by differential equations that depending on properties such ad the heat capacity, heat absorption, thermal resistance

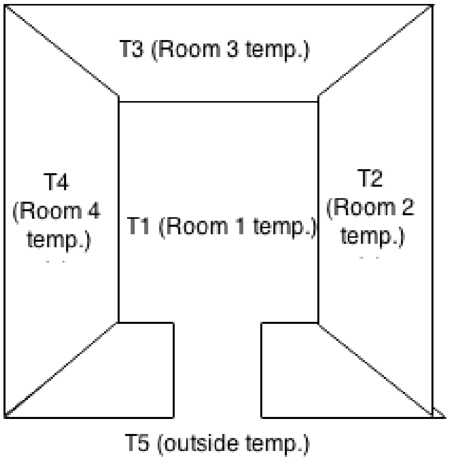


Figure 1: Building layout for HVAC control. We show results for the temperature in Room 1.

and area of the walls between the room and its neighboring rooms, the radiative heat flux density on external walls, heat capacity and air mass flow into the room, transmissivity of the glass of windows, and the total area of the windows on walls surrounding the room, and the internal heat generation in the room. Further details on this thermal model can be found in [17].

The heat transfer equations for each wall and room yield the following system dynamics:

$$\dot{x}_t = f(x_t, u_t, d_t), \quad y_t = Cx_t,$$

where $x_t \in \mathbb{R}^n$ is the state vector representing the temperature of the nodes in the thermal network (including rooms and walls), and $u_t \in \mathbb{R}^{lm}$ is the input vector representing the air mass flow rate and discharge air temperature of conditioned air into each thermal zone (with l being the number of inputs to each thermal zone, e.g. air mass flow and supply air temperature). The vector d_t stores the estimated disturbance values, aggregating various unmodeled dynamics such as the outside temperature, internal heat generation and radiative heat flux density, and can be estimated using historical data [17]. In this work, we show results for controlling the temperature of Room 1 only, and include the temperature of the neighboring rooms as part of these unmodeled dynamics d_t. $y_t \in \mathbb{R}^m$ is the output vector, representing the temperature of the thermal zones, and C is a constant matrix of proper dimension.

Assume that the system dynamics are discretized with a sampling time of τ, and let H be the prediction horizon (in number of time steps). Here we consider $\tau = 0.5$ hr and $H = 12$. At each time t, the receding horizon controller solves an optimal control problem to compute $\vec{u}_t = [u_t, \ldots, u_{t+H-1}]$, minimizing the cumulative norm of u_t: $\sum_{k=0}^{H-1} \|u_{t+k}\|$. We assume known an occupancy function occ_t, which is equal to 1 when the room is occupied and to 0 otherwise. The purpose of the controller is to maintain a comfort temperature given by T^{comf} whenever the room is occupied, while minimizing the cost of heating. The assumption on the environment is that the disturbances d_t are in a range bounded by ϵ around some reference d_t^{ref}, obtained from historical data. Formally,

Here

$$\begin{aligned}
&x_{t+1} = f(x_t, u_t, d_t)\\
&\varphi_e = \Box_{[0,H]}(|d_t - d_t^{\text{ref}}| < \epsilon)\\
&\varphi_s = \Box_{[0,H]}((\text{occ}_t > 0) \Rightarrow (T_t > T_t^{\text{comf}})\\
&J(\mathbf{f}(x_0, \mathbf{u}^H, \mathbf{w}^H)) = \textstyle\sum_{k=0}^{H-1} \|u_{t+k}\|
\end{aligned}$$

The STL formula φ was encoded using the robust MILP encoding. Figure 6.1 presents results of executing the receding horizon controller synthesized using Algorithm 1, while modeling the disturbance as bounded (more precisely, satisfying φ_e) but non-deterministic; we used $\epsilon = 3$. Compare this with Figure 6.1, where the disturbance is modeled as corresponding exactly to d_t^{ref} (i.e. $\epsilon = 0$). In both cases, the actual disturbance (undepicted) was exactly d^{ref}.

We observe that the controller designed to operate in an adversarial environment is more conservative, and starts heating the room earlier (e.g. time step 4 instead of 5) in response to the same predicted occupancy signal, to account for the possibility of a higher disturbance. Additionally, when the occupancy signal is non-zero, the control input applied to counter the worst case disturbance results in a temperature that is higher than in the deterministic case; the result is that the temperature plot rises further above the minimum temperature of T_{comf} in the nondeterministic case.

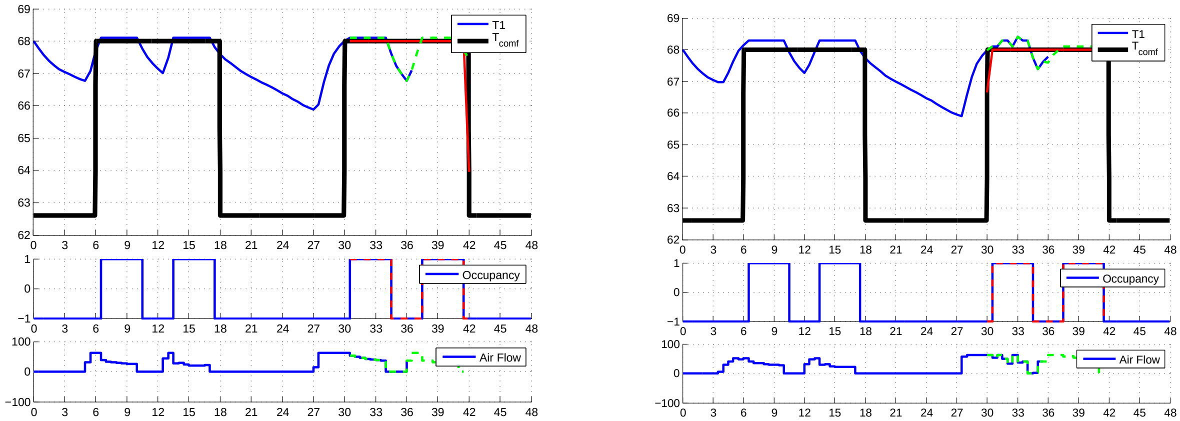
As we previously observed in [21], most of the time is spent initially creating the MILP, while solving it takes a fraction of a second for each time step. In practice, the CEGIS loop of Algorithm 1 was executed fewer than 2 times for most time steps. As expected, the number of CEGIS iterations was greater for the case where $\epsilon = 3$ than $\epsilon = 0$, reflecting the greater nondeterminism.

The HVAC model used in this case study is 5-dimensional [17]; this represents a significant improvement over reactive synthesis techniques based on discrete abstraction, which do not typically scale past 2 or 3 continuous variables. We expect our techniques to scale well to higher dimensions. The main culprit when it comes to problem size is the length of the horizon required to ensure satisfiability. This increases with the nesting of temporal operators.

6.2 Autonomous Driving in Nondeterministic Environments

We now consider the problem of controlling an autonomous vehicle operating in the presence of other, potentially adversarial vehicles.

In this example, two moving vehicles approach an intersection, which they must cross. We let the red car in Figure 3 be the *ego* vehicle (the vehicle we control), and the black car be part of the environment. We define the state space using a simplified 6-dimensional model, with the position of the two vehicles $((x^{\text{ego}}, y^{\text{ego}}), (x^{\text{adv}}, y^{\text{adv}}))$ and the velocity of the two $(v^{\text{ego}} = \dot{y}^{\text{ego}}, v^{\text{adv}} = \dot{x}^{\text{adv}})$ in ms^{-1} as state vari-



(a) Known Disturbances: $\epsilon = 0$

(b) Uncertain Disturbances: $\epsilon = 3$

Figure 2: Receding horizon control of Room 1 temperature under constraints based on occupancy, expressed in STL. In both plots, the black line in the topmost subplots represent the actual value of T_{comf}, and the red line is the prediction of T_{comf} for the current planning horizon. The blue line in the plot of the occupancy signal represents the actual value, and the red dotted overlay is the prediction for the current horizon. The green dotted lines in the temperature and air flow plots depict the control input and resulting state trajectory computed during the current iteration of the receding horizon control computation. The blue lines in the control and temperature plots represent the portion of the control input that was actually executed and the resulting temperature T_1, respectively.

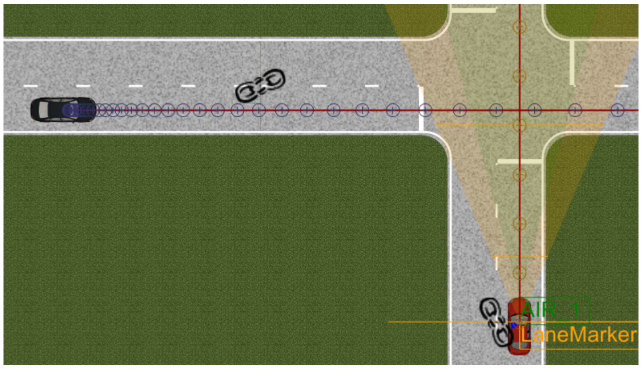


Figure 3: Two vehicles crossing an intersection simultaneously. The red car is the *ego* vehicle, which we control. The black car is part of the adversarial environment.

ables, and the acceleration ($a^{\text{ego}} = \dot{v}^{\text{ego}}$) of the ego vehicle as a single input. The disturbance is the acceleration of the environment vehicle ($a^{\text{adv}} = \dot{v}^{\text{adv}}$), which is allowed to take values in a bounded range. Thus:

$$\mathbf{x}_t = \begin{bmatrix} x_t^{\text{ego}} & y_t^{\text{ego}} & x_t^{\text{adv}} & y_t^{\text{adv}} & v_t^{\text{ego}} & v_t^{\text{adv}} \end{bmatrix}^\top \tag{2}$$

$$\mathbf{u} = a_t^{\text{ego}} \quad \mathbf{w} = a_t^{\text{adv}} \tag{3}$$

We assume each vehicle has the dynamics of a double integrator:

$$\begin{bmatrix} \dot{x}^{\text{ego}} \\ \dot{y}^{\text{ego}} \\ \dot{v}^{\text{ego}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{\text{ego}} \\ y^{\text{ego}} \\ v^{\text{ego}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u} \tag{4}$$

$$\begin{bmatrix} \dot{x}^{\text{adv}} \\ \dot{y}^{\text{adv}} \\ \dot{v}^{\text{adv}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{\text{adv}} \\ y^{\text{adv}} \\ v^{\text{adv}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{w} \tag{5}$$

Our specification in this example is that there should be no collision at the intersection for the two vehicles, and that the ego vehicle's speed should be close to $1ms^{-1}$. Here the disturbance $\mathbf{w}$ is the acceleration of the adversary, whose value is assumed to be close to a reference value, $\mathbf{w}^{\text{ref}}$. We use the following STL formulas:

$$\varphi_e = \square(|\mathbf{w} - \mathbf{w}^{\text{ref}}| < 0.1)$$
$$\varphi_s = \square(|y_t^{\text{ego}} - x_t^{\text{adv}}| < 2) \implies \square_{[0,2]}(|v_t^{\text{ego}}| < 0.1)$$

The formula φ_s specifies that whenever y_t^{ego} is close to x_t^{adv}, i.e. within the range of $2m$, the ego vehicle should come to a stop ($|v_t^{\text{ego}}| < 0.1$) for a short period of time (2s). Figure 3, shows that the two vehicles will be close only when they are in the vicinity of the intersection. We expect the ego vehicle to stop at the intersection in order to allow the adversary to cross. In addition, we optimize the following cost function, which encourages the ego vehicle's speed to be close to $1ms^{-1}$.

$$J(\mathbf{f}(x_0, \mathbf{u}^H, \mathbf{w}^H)) = \sum_{k=0}^{H-1} ||v_{t+k}^{\text{ego}} - 1|| \qquad (6)$$

Figure 4 illustrates the result of applying Algorithm 2 to synthesize control inputs for the ego vehicle. The first plot shows the position of the two vehicles, x_t^{adv} and y_t^{ego} (in m). The ego vehicle starts with a negative value on its y-axis $y_0^{\text{ego}} < 0$, and the adversary starts with a positive x-value $x_0^{\text{adv}} > 0$. Here the origin represents the middle of the intersection: at any time t if $y_t^{\text{ego}} = x_t^{\text{adv}} = 0$, the two cars have collided. The synthesized control input should therefore avoid such a collision, and the two vehicles should not be at location 0 or its vicinity ($|y_t^{\text{ego}} - x_t^{\text{adv}}| < 2$) at the same time.

As seen in the first and second subplots in Figure 4, at time $t = 8s$, the ego vehicle stops at its current position in order to avoid collision with the adversary car. The vehicle proceeds after a short stop to let the adversary pass. The third subplot shows the velocity of the two vehicles, and the fourth plot represents the acceleration. Notice that the velocity of the ego vehicle stabilizes at $1ms^{-1}$ at most times as long as it avoids any collisions. The accelerations shown in the fourth plot include the control input synthesized using Algorithm 2, and the disturbance, i.e., the acceleration of the adversary.

7. DISCUSSION

The main contribution of this paper is a CEGIS procedure for synthesis of reactive controllers for systems satisfying STL specifications. We showed how our approach can be used as part of a receding horizon control scheme, to generate control for systems that must satisfy STL properties in the presence of adversarial environments, subject to domain-specific cost functions. We presented experimental results for controller synthesis on simplified models of a smart-building HVAC system and an autonomous car, and showed in simulation that the synthesized controllers satisfy the specified properties despite nondeterministic and adversarial environments.

8. ACKNOWLEDGEMENTS

This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

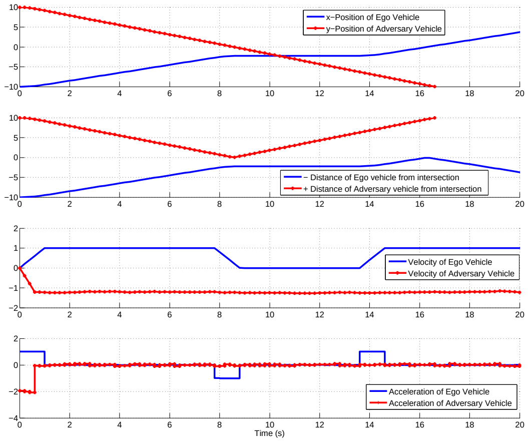


Figure 4: Plot of position, velocity and acceleration of the ego and adversary vehicles. The second plot from the top shows the distance (in m) of each vehicle from the intersection. While the adversary vehicle drives straight through the intersection at a constant speed, the ego vehicle stops at t=8s, when it is around 2.5m from the intersection, then resumes moving around t=16s, thus avoiding collision.

9. REFERENCES

[1] R. Alur, R. Bodik, G. Juniwal, M. M. K. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. In *Proceedings of the IEEE International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, October 2013.

[2] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[3] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS*, pages 193–207, 1999.

[4] K. Chatterjee, T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided planning. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 104–111, 2005.

[5] X. C. Ding, M. Lazar, and C. Belta. LTL receding horizon control for finite deterministic systems. *Automatica*, 50(2):399–408, 2014.

[6] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS*, pages 92–106, 2010.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343 – 352, 2009.

[8] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

[9] E. A. Gol and M. Lazar. Temporal logic model predictive control for discrete-time systems. In *Proceedings of the 16th international conference on*

Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA, pages 343–352, 2013.

[10] X. Jin, A. Donzé, J. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *HSCC'13*, 2013.

[11] S. Karaman and E. Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, México*, pages 3953–3958, 2008.

[12] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009, combined withe the 28th Chinese Control Conference, December 16-18, 2009, Shanghai, China*, pages 2222–2229, 2009.

[13] S. Karaman and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control*, 21(12):1372–1395, 2011.

[14] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transaction on Automatic Control*, 53(1):287–297, 2008.

[15] Y. Kwon and G. Agha. Ltlc: Linear temporal logic for control. In M. Egerstedt and B. Mishra, editors, *HSCC*, pages 316–329, 2008.

[16] M. Maasoumy, C. Rosenberg, A. Sangiovanni-Vincentelli, and D. Callaway. Model predictive control approach to online computation of demand-side flexibility of commercial buildings hvac systems for supply following. In *IEEE American Control Conference (ACC 2014)*, Portland, USA, June 2014.

[17] M. Maasoumy Haghighi. *Controlling Energy-Efficient Buildings in the Context of Smart Grid: A Cyber Physical System Approach*. PhD thesis, University of California, Berkeley, Dec 2013.

[18] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.

[19] R. M. Murray, J. Hauser, A. Jadbabaie, M. B. Milam, N. Petit, W. B. Dunbar, and R. Franz. Online control customization via optimization-based control. In *In Software-Enabled Control: Information Technology for Dynamical Systems*, pages 149–174. Wiley-Interscience, 2002.

[20] P. Nuzzo, H. Xu, N. Ozay, J. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia. A contract-based methodology for aircraft electric power system design. *Access, IEEE*, PP(99):1–1, 2013.

[21] V. Raman, M. Maasoumy, A. Donzé, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *Proc. of the IEEE Conf. on Decision and Control*, 2014.

[22] A. Solar-Lezama, L. Tancau, R. Bodík, S. A. Seshia, and V. A. Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 404–415. ACM Press, October 2006.

[23] E. M. Wolff, U. Topcu, and R. M. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 5319–5325, 2014.

[24] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE Trans. Automat. Contr.*, 57(11):2817–2830, 2012.