Submitted, IEEE T. Automatic Control (2 Jan 2016)
http://www.cds.caltech.edu/~murray/preprints/ram+16-tac_s.pdf

Model Predictive Control for Signal Temporal Logic Specifications

Vasumathi Raman[1], Alexandre Donzé[2], Mehdi Maasoumy[3],
Richard M. Murray[1], Alberto Sangiovanni-Vincentelli[2] and Sanjit A. Seshia[2]

***Abstract*—We present a mathematical programming-based method for model predictive control of cyber-physical systems subject to signal temporal logic (STL) specifications. We describe the use of STL to specify a wide range of properties of these systems, including safety, response and bounded liveness. For synthesis, we encode STL specifications as mixed integer-linear constraints on the system variables in the optimization problem at each step of a receding horizon control framework. We prove correctness of our algorithms, and present experimental results for controller synthesis for building energy and climate control.**

***Index Terms*—formal synthesis, timed logics, model predictive control, cyberphysical systems**

I. Introduction

Controlling a cyber-physical system (CPS) involves handling complex interactions between computing components and their physical environment, and often necessitates hierarchies of controllers. Typically at the highest level, a *supervisory* controller is responsible for making high-level decisions, while at the lowest level traditional control laws such as PID control are used. In general, the design of these different controllers is done mostly in isolation at each level, and their combination is implemented ad hoc. As the complexity of these systems grows, reasoning about the correctness of interactions between the various layers of control becomes increasingly challenging, begging automation.

Formal methods is the subfield of computer science concerned with verification and synthesis, i.e., automatic and rigorous design of digital systems. It provides mathematical formalisms for specifying behaviors and algorithms for verification and synthesis of a system against properties specified within these formalisms. Methods for synthesis of correct-by-construction discrete supervisory controllers have been developed and successfully used for cyber-physical systems in domains including robotics [1] and aircraft power system design [2]. However, for physical systems that require constraints not just on the order of events, but on the temporal distance between them, simulation and testing is still the method of choice for validating properties and establishing guarantees; the exact exhaustive verification or synthesis of such systems is in general undecidable [3].

Model predictive control (MPC) or receding horizon control (RHC) is based on iterative, finite horizon optimization over a model of the *plant*, i.e. the system to be controlled. At any given time t, the current plant state is observed, and an optimal control strategy computed for some finite time horizon in the future, $[t, t+H]$. An online calculation is used to explore possible future state trajectories originating from the current state, finding an optimal control strategy until time $t+H$. To ensure robustness with respect to model errors, only the first step of the computed optimal control strategy is implemented; then the plant state is sampled again, and new calculations are performed on a horizon of H starting from the new current state. While the global optimality of such receding horizon approach is not ensured, it tends to do well in practice. In addition to reducing computational complexity, it improves the system robustness with respect to exogenous disturbances and modeling uncertainties [4]. Another reason Model Predictive Control (MPC) is particularly attractive to industry is its ability to handle constrained dynamical systems [5].

Signal temporal logic (STL) [6] was originally developed in order to specify and monitor the expected behavior of physical systems, including temporal constraints between events. Signal Temporal Logic (STL) allows the specification of properties of dense-time, real-valued signals, and the automatic generation of monitors for testing these properties on individual simulation traces. It has since been applied to the analysis of several types of continuous and hybrid systems, including dynamical systems and analog circuits, where the continuous variables represent quantities like currents and voltages in a circuit. STL has the advantage of naturally admitting *quantitative* semantics which, in addition to the yes/no answer to the satisfaction question, provide a real number that grades the quality of the satisfaction or violation. Such semantics have been defined for timed logics, including Metric Temporal Logic (MTL) [7] and STL [8], to assess the *robustness* of systems to parameter or timing variations.

In this paper, we solve the problem of control synthesis from STL specifications using a receding horizon approach. We allow the user to specify desired properties of the system using an STL formula, and synthesize control such that the system satisfies that specification, while using a receding horizon approach to ensure practicality and robustness. We do so by

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

[1]V. Raman and R. M. Murray are with the California Institute of Technology, Pasadena, CA 91125, USA vasu@caltech.edu, murray@cds.caltech.edu

[2]A. Donzé, A. Sangiovanni-Vincentelli and S. A. Seshia are with the Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA 94720, USA donze@berkeley.edu, alberto@berkeley.edu, sseshia@eecs.berkeley.edu

[3]M. Maasoumy is with C3 Energy Inc. Redwood City, CA 94063, USA mehdi.maasoumy@c3energy.com

Manuscript received January 2, 2015.

decomposing the STL specifications into a series of formulas over each time horizon, such that synthesizing a controller fulfilling the formula at each horizon results in satisfaction of the global specification. Recent work on optimal control synthesis of aircraft load management systems [9] represented STL-like specifications as time-dependent equality and inequality constraints, yielding a Mixed Integer Linear Program (MILP). The MILP was then solved in an MPC framework, yielding an optimal control policy. However, the manual transformation of specifications into equality and inequality constraints is cumbersome and problem-specific. As a key contribution, this paper presents two *automatically-generated* MILP encodings for such STL specifications.

Our main contribution is a pair of Bounded Model Checking (BMC)-style encodings [10] for STL specifications as MILP constraints on a cyber-physical system. We show how these encodings can be used to generate open-loop control signals that satisfy finite and infinite horizon STL properties and, moreover, to generate signals that maximize quantitative (robust) satisfaction. We provide a fragment of STL, denoted SNN-STL, such that, under reasonable assumptions on the system dynamics, the problem of synthesizing an open-loop control sequence such that the system satisfies a provided specification is a Linear Program (LP), and is therefore polynomial-time solvable. We also demonstrate how our MILP formulation of the STL synthesis problem can be used in an MPC framework to compute feasible and optimal controllers for cyber-physical systems under timed specifications. We present experimental results comparing both encodings, and two case studies: one on a thermal model of an Heating Ventilation and Air Conditioning (HVAC) system, and another in the context of regulation services in a micro-grid. These case studies were previously reported in [11], [12]. We show how the MPC schemes in these examples can be framed in terms of synthesis from an STL specification, and present simulation results to illustrate the effectiveness of our methodology.

II. Preliminaries

A. Systems

We consider a continuous-time system Σ of the form

$$\dot{x}_t = f(x_t, u_t, w_t)$$

where $x_t \in \mathcal{X} \subseteq (\mathbb{R}^{n_c} \times \{0,1\}^{n_l})$ are the continuous and binary/logical states, $u_t \in U \subseteq (\mathbb{R}^{m_c} \times \{0,1\}^{m_l})$ are the (continuous and logical) control inputs, $w_t \in W \subseteq (\mathbb{R}^{e_c} \times \{0,1\}^{e_l})$ are the external environment inputs (also referred to as "disturbances"). As x_t includes binary components, the above ODE may contain discontinuities corresponding to switches in these components. In general, such hybrid systems are more accurately modeled using differential-algebraic equations, but we do not dwell on this point since we approximate their behavior with a difference equation, as follows.

Given a sampling time $\Delta t > 0$, we assume that Σ admits a discrete-time approximation Σ_d of the form

$$x(t_{k+1}) = f_d(x(t_k), u(t_k), w(t_k)) \tag{1}$$

where for all $k > 0$, $t_{k+1} - t_k = \Delta t$. A *run* of Σ_d is a sequence

$$\xi = (x_0 u_0 w_0)(x_1 u_1 w_1)(x_2 u_2 w_2)...$$

where $x_k = x(t_k) \in \mathcal{X}$ is the state of the system at index k, and for each $k \in \mathbb{N}$, $u_k = u(t_k) \in U$, $w_k = w(t_k) \in W$ and $x_{k+1} = f_d(x_k, u_k, w_k)$. We assume that given an initial state $x_0 \in X$, a control input sequence $\mathbf{u}^N = u_0 u_1 u_2 \dots u_{N-1} \in U^N$, and a sequence of environment inputs $\mathbf{w}^N = w_0 w_1 w_2 \dots w_{N-1} \in W^N$, the resulting horizon-$N$ run of a system modeled by equation (1), which we denote by

$$\xi(x_0, \mathbf{u}^N, \mathbf{w}^N) = (x_0 u_0 w_0)(x_1 u_1 w_1)(x_2 u_2 w_2)...(x_N u_N w_N),$$

is unique. In addition, we introduce a generic cost function $J(\xi(x_0, \mathbf{u}, \mathbf{w}))$ that maps (infinite and finite) runs to $\mathbb{R}$.

B. Signal Temporal Logic

We consider STL formulas defined recursively according to the following grammar:

$$\varphi ::= \pi^\mu \mid \neg\psi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_{[a,b]}\ \varphi \mid \varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2,$$

where π^μ is an atomic predicate $\mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathbb{B}$ whose truth value is determined by the sign of a function $\mu : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}$ and φ_1, φ_2 are STL formulas. The fact that a run $\xi(x_0, \mathbf{u}, \mathbf{w})$ satisfies an STL formula φ is denoted by $\xi \models \varphi$. Informally, $\xi \models \Diamond_{[a,b]}\, \varphi$ if φ holds at some time step between a and b, and $\xi \models \varphi\ \mathcal{U}_{[a,b]}\ \psi$ if φ holds at every time step before ψ holds, and ψ holds at some time step between a and b. Additionally, we define $\Box_{[a,b]}\, \varphi = \neg \Diamond_{[a,b]} (\neg\varphi)$, so that $\xi \models \Box_{[a,b]}\, \varphi$ if φ holds at *all* times between a and b. Formally, the validity of a formula φ with respect to the run ξ is defined inductively as follows

$$\begin{array}{lcl}
\xi \models \varphi & \Leftrightarrow & (\xi, t_0) \models \varphi \\
(\xi, t_k) \models \pi^\mu & \Leftrightarrow & \mu(x_k, y_k, u_k, w_k) > 0 \\
(\xi, t_k) \models \neg\psi & \Leftrightarrow & \neg(\xi, t_k) \models \psi) \\
(\xi, t_k) \models \varphi \wedge \psi & \Leftrightarrow & (\xi, t_k) \models \varphi \wedge (\xi, t_k) \models \psi \\
(\xi, t_k) \models \Diamond_{[a,b]}\, \varphi & \Leftrightarrow & \exists t_{k'} \in [t_k + a, t_k + b], (\xi, t_{k'}) \models \varphi \\
(\xi, t_k) \models \varphi \mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t_{k'} \in [t_k + a, t_k + b] \text{ s.t. } (\xi, t_{k'}) \models \psi \\
 & & \wedge \forall t_{k''} \in [t_k, t_{k'}], (\xi, t_{k''}) \models \varphi.
\end{array}$$

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 such that $t_N \geq 10 + 6 = 16$ is sufficient to determine whether the formula is satisfiable.

Remark 1. Here we have defined a semantics for STL over discrete-time signals, which is formally equivalent to the simpler Linear Temporal Logic (LTL), once time and predicates are abstracted into steps and Boolean variables, respectively. There are several advantages of still using STL over LTL, though. First, STL allows us to explicitly use real time in our specifications instead of abstract integer indices, which improves the readability relative to the original system's behaviors. Second, although in the rest of this paper we focus on the control of the discrete-time system Σ_d, our goal is to use the resulting controller for the control of the continuous

system Σ. Hence the specifications should be independent from the sampling time Δt. Finally, note that the relationship between the continuous-time and discrete-time semantics of STL, depending on discretization error and sampling time, is beyond the scope of this paper. The interested reader can refer to [13] for further discussion on this topic.

C. Quantitative semantics for STL

Quantitative or robust semantics for STL are defined by providing a real-valued function ρ^{φ} of signal ξ and time t such that $\rho^{\varphi}(\xi,t) > 0 \Rightarrow (\xi,t) \models \varphi$. We define one such function recursively, as follows:

$$\begin{aligned}
\rho^{\pi^{\mu}}(\xi,t_k) &= \mu(x_k,y_k,u_k,w_k)\\
\rho^{\neg\psi}(\xi,t_k) &= -\rho^{\psi}(\xi,t_k)\\
\rho^{\varphi_1\wedge\varphi_2}(\xi,t_k) &= \min(\rho^{\varphi_1}(\xi,t_k),\rho^{\varphi_2}(\xi,t_k))\\
\rho^{\varphi_1\vee\varphi_2}(\xi,t_k) &= \max(\rho^{\varphi_1}(\xi,t_k),\rho^{\varphi_2}(\xi,t_k))\\
\rho^{\Diamond_{[a,b]}\,\psi}(\xi,t_k) &= \max_{t_{k'}\in[t+a,t+b]}\rho^{\psi}(\xi,t_{k'})\\
\rho^{\varphi_1\,\mathcal{U}_{[a,b]}\,\varphi_2}(\xi,t_k) &= \max_{t_{k'}\in[t+a,t+b]}(\min(\rho^{\varphi_2}(\xi,t_{k'}),\\
&\qquad \min_{t_{k''}\in[t_k,t_{k'}]}\rho^{\varphi_1}(\xi,t_{k''}))
\end{aligned}$$

Note that if $\mu(x_k,y_k,u_k,w_k)=0$, neither $\rho^{\pi^{\mu}}(\xi,t_k)>0$ nor $\rho^{\pi^{\mu}}(\xi,t_k)>0$. Therefore, $(\xi,t)\models\varphi \not\Rightarrow \rho^{\varphi}(\xi,t)>0$. To simplify notation, we denote $\rho^{\pi^{\mu}}$ by ρ^{μ} for the remainder of the paper. The robustness of satisfaction for an arbitrary STL formula is computed recursively from the above semantics by propagating the values of the functions associated with each operand using $\min$ and $\max$ operators corresponding to the various STL operators. For example, the robust satisfaction of π^{μ_1} where $\mu_1(x)=x-3>0$ at time 0 is $\rho^{\mu_1}(\xi,0)=x_0-3$. The robust satisfaction of $\mu_1\wedge\mu_2$ is the minimum of ρ^{μ_1} and ρ^{μ_2}. Temporal operators are treated as conjunctions and disjunctions along the time axis: since we deal with discrete time, the robustness of satisfaction of $\varphi=\Box_{[0,2.1]}\,\mu_1$ is

$$\begin{aligned}
\rho^{\varphi}(x,0) &= \min_{t_k\in[0,2.1]}\rho^{\mu_1}(x,t_k)\\
&= \min\{x_0-3,x_1-3,\ldots,x_K-3\},
\end{aligned}$$

where $0\le t_0<t_1<\ldots<t_K\le 2.1<t_{K+1}$.

The robustness score $\rho^{\varphi}(\xi,t)$ can be interpreted as *how much* ξ satisfies φ. Its absolute value can be viewed as the signed distance of ξ from the set of trajectories satisfying or violating φ, in the space of projections with respect to the function μ that define the predicates of φ [7].

III. Problem Statement

We now formally state the STL control synthesis problem and its model predictive control formulation. Given an STL formula φ, a *cost function* of the form $J(x_0,\mathbf{u},\mathbf{w},\varphi)\in\mathbb{R}$, an initial state $x_0\in\mathcal{X}$, a horizon L and a reference disturbance signal $\mathbf{w}\in\mathcal{W}^N$, we formulate two problems: *open-loop* and *closed-loop* synthesis. The two scenarios are depicted as block diagrams in Fig. 1 and Fig. 2.

Problem 1 (open-loop)**.** Compute $\mathbf{u}^*=u_0^*u_1^*\ldots u_{N-1}^*$ where

$$\begin{aligned}
\mathbf{u}^* = \quad & \underset{\mathbf{u}\in\mathcal{U}^N}{\operatorname{argmin}}\, J(x_0,\mathbf{u},\mathbf{w},\varphi)\\
& \text{s.t. } \xi(x_0,\mathbf{u},\mathbf{w})\models\varphi
\end{aligned}$$

Note that we assume that the state of the plant is fully observable, and the environment inputs are known in advance.

Problem 2 (closed-loop)**.** Given a horizon $0<L<N$, for all $0\le k\le N-L$, compute $u_k^*=u_k^{L*}$, the first element of the sequence $\mathbf{u}_k^{L*}=u_k^{L*}u_{k+1}^{L*}\ldots u_{k+L-1}^{L*}$ satisfying

$$\begin{aligned}
\mathbf{u}_k^{L*} = \quad & \underset{\mathbf{u}_K^L\in\mathcal{U}^L}{\operatorname{argmin}}\, J(x_k,\mathbf{u}_k^L,\mathbf{w}_k,\varphi)\\
& \text{s.t. } \xi(x_k,\mathbf{u}_k^L,\mathbf{w}_k)\models\varphi
\end{aligned}$$

The closed-loop formulation corresponds to a model predictive control scheme, where the reference disturbance can change at each iteration k.

In Sections IV and V, we present both an open-loop solution to Problem 1, and a solution to Problem 2 for a large class of STL formulas. In the absence of an objective function J on runs of the system, we maximize the robustness of the generated runs with respect to φ. A key component of our solution is encoding the STL specifications as MILP constraints, which can be combined with MILP constraints representing the system dynamics to efficiently solve the resulting state-constrained optimization problem.

IV. Open-loop Controller Synthesis

To solve Problem 1, we extend the bounded model checking encoding of [14] from finite, discrete systems to dynamical systems using mixed-integer programming instead of SAT. Our presentation and notation below follow that of [15]. For open-loop controller synthesis, we will search for a trajectory of length N that satisfies φ. To admit STL formulas describing infinite runs, we parametrize an infinite sequence of states using a finite sequence with a loop. Imposing this lasso-shaped structure renders our synthesis procedure conservative for infinite-state systems, in the sense that a solution may exist that is not found when imposing such a structure. For finite-state systems, the lasso shape is without loss of generality but we must still find an appropriate trajectory length N.

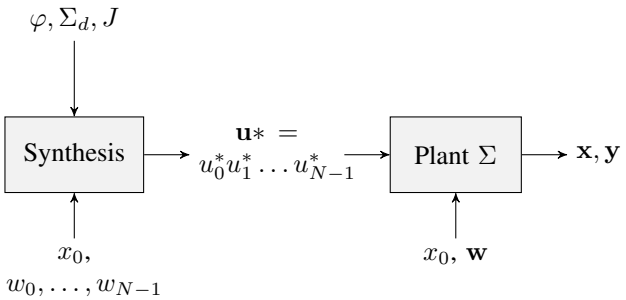

Fig. 1. Open-loop problem formulation: given an STL formula ϕ, the discrete-time plant model Σ_d, the cost function J, the goal is to generate a sequence of control inputs $\mathbf{u}^*$ over a horizon of N time steps. The problem is additionally parametrized by the the initial state x_0 and disturbance vector $\mathbf{w}$.

The encoding of Problem 1 as an MILP consists of system constraints, loop constraints and STL constraints, as defined below.

A. Constraints on system evolution

The first component of the set of constraints is provided by the system model. Our approach applies to any system that yields to a MILP formulation for model predictive control over horizon N. The system constraints encode valid finite (horizon-N) trajectories for a system with form (1) – these constraints hold if and only if the trajectory $\mathbf{x}(x_0, \mathbf{u}_N)$ satisfies (1) for $t = 0, 1, ..., N$. Note that this is quite general, and accommodates any system for which the resulting constraints and objectives form a mixed integer-linear program. An example is the smart grid regulation control system presented in [16]. Other useful examples include mixed logical dynamical systems such as those presented in [17]. Other cost functions and system dynamics can also be included by using appropriate solvers.

B. Loop constraints for trajectory parametrization

As described above, to synthesize open-loop control for unbounded (infinite-horizon) specifications, we parametrize the trajectory as a lasso, i.e. constrain it to contain a loop. This loop encoding is again inspired by the basic idea of bounded model checking [10], which is to consider only a finite prefix of a path when looking for a solution to an existential model checking problem. A crucial observation is that although the considered path prefix is finite, it can still represent an infinite path if there is a loop back from the last state to any of the previous states.

To enforce the existence of a loop in the finite system trajectory, we introduce N binary variables $l_1, ..., l_N$, which determine where the loop forms. These are constrained such that only one can be high at a time, and if $l_j = 1$, then $x_{j-1} = x_N$. The following constraints enforce these requirements:

- $\sum_{j=1}^{N} l_j = 1$
- $x_N \leq x_{j-1} + M_j(1 - l_j),\ j = 1, ..., N,$
- $x_N \geq x_{j-1} + M_j(1 - l_j),\ j = 1, ..., N,$

where M_j are sufficiently large positive numbers, picked based on $\mathcal{X}$.

C. Boolean encoding of STL constraints

Given a formula φ, we introduce a variable z_t^φ, whose value is tied to a set of mixed integer linear constraints required for the satisfaction of φ at position t in the state sequence of horizon N. In other words, z_t^φ has an associated set of MILP constraints such that $z_t^\varphi = 1$ if and only if φ holds at position t. We recursively generate the MILP constraints corresponding to z_0^φ – the value of this variable determines whether a formula φ holds in the initial state.

1) Predicates: The predicates are represented by constraints on system state variables. For each predicate $\mu \in P$, we introduce binary variables $z_t^\mu \in \{0, 1\}$ for times $t = 0, 1, ..., N$. The following constraints enforce that $z_t^\mu = 1$ if and only if $\mu(x_t) > 0$:

$$\begin{array}{rcl} \mu(x_t) & \leq & M_t z_t^\mu - \epsilon_t \\ -\mu(x_t) & \leq & M_t(1 - z_t^\mu) - \epsilon_t \end{array}$$

where M_t are sufficiently large positive numbers, and ϵ_t are sufficiently small positive numbers that serve to bound $\mu(x_t)$ away from 0. This encoding restricts the set of STL formulas that can be encoded using our approach to those over linear predicates, but admits arbitrary STL formulas over such predicates.

2) Boolean operations on MILP variables: As described in Section IV-C1, each predicate μ has an associated binary variable z_t^μ which equals 1 if μ holds at time t, and 0 otherwise. In fact, by the recursive definition of our MILP constraints on STL formulas, each operand φ in a Boolean operation has a corresponding variable z_t^φ which is 1 if φ holds at t and 0 otherwise. Here we define Boolean operations on these variables: these are the building blocks of our recursive encoding. The definitions in this subsection are consistent with those in [15].

Logical operations on variables $z_t^\psi \in [0, 1]$ are defined as follows:

Negation: $z_t^\psi = \neg z_t^\varphi$ $\qquad z_t^\psi = 1 - z_t^\varphi$

Conjunction: $z_t^\psi = \wedge_{i=1}^m z_{t_i}^{\varphi_i}$ $\qquad \begin{array}{l} z_t^\psi \leq z_{t_i}^{\varphi_i}, i = 1, ..., m, \\ z_t^\psi \geq 1 - m + \sum_{i=1}^m z_{t_i}^{\varphi_i} \end{array}$

Disjunction: $z_t^\psi = \vee_{i=1}^m z_{t_i}^{\varphi_i}$ $\qquad \begin{array}{l} z_t^\psi \geq z_{t_i}^{\varphi_i}, i = 1, ..., m, \\ z_t^\psi \leq \sum_{i=1}^m z_{t_i}^{\varphi_i} \end{array}$

Given a formula ψ containing a Boolean operation, we add new continuous variables $z_t^\psi \in [0, 1]$, and set $z_t^\psi = \neg z_t^\mu$, $z_t^\psi = \wedge_{i=1}^m z_{t_i}^{\varphi_i}$, and $z_t^\psi = \vee_{i=1}^m z_{t_i}^{\varphi_i}$ for $\psi = \neg\mu$, $\psi = \wedge_{i=1}^m \varphi_i$ and $\psi = \vee_{i=1}^m \varphi_i$, respectively. These constraints enforce that $z_t^\psi = 1$ if ψ holds at time t and $z_t^\psi = 0$ otherwise.

3) Temporal constraints: We first present encodings for the $\Box$ and $\Diamond$ operators. We will use these encodings to define the encoding for the $\mathcal{U}_{[a,b]}$ operator.

Always: $\psi = \Box_{[a,b]}\, \varphi$

Let $a_t^N = \min(t + a, N)$ and $b_t^N = \min(t + b, N)$

Define $z_t^\psi = \wedge_{i=a_t^N}^{b_t^N} z_i^\varphi \wedge (\bigvee_{j=1}^N l_j \wedge \bigwedge_{i=\hat{a}_j^N}^{\hat{b}_j^N} z_i^\varphi)$

The logical operation $\wedge$ on the variables z_i^φ here is as defined in Section IV-C2. Intuitively, this encoding enforces that the formula φ is satisfied at every time step on the interval $[a, b]$ relative to time step t.

Eventually: $\psi = \Diamond_{[a,b]}\, \varphi$

Define $z_t^\psi = \vee_{i=a_t^N}^{b_t^N} z_i^\varphi \wedge (\bigvee_{j=1}^N l_j \wedge \bigvee_{i=\hat{a}_j^N}^{\hat{b}_j^N} z_i^\varphi)$ This encoding enforces that the formula φ is satisfied at some time step on the interval $[a, b]$ relative to time step t.

Until: $\psi = \varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2$

The bounded until operator $\mathcal{U}_{[a,b]}$ can be defined in terms of the unbounded $\mathcal{U}$ (inherited from LTL) as follows [18]:

$$\varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2 = \Box_{[0,a]}\, \varphi_1 \wedge \Diamond_{[a,b]}\, \varphi_2 \wedge \Diamond_{[a,a]}(\varphi_1\ \mathcal{U}\ \varphi_2)$$

We will use the encoding of the unbounded $\mathcal{U}$ from [10]. When encoding over infinite trajectories, this requires

an auxiliary encoding that prevents the pitfalls of circular reasoning on the finite parametrization of the infinite sequences. The interested reader is referred to [10] for the details of the encoding. The auxiliary encoding of the unbounded until is

$$\langle\langle\, \varphi_1 \,\mathcal{U}\, \varphi_2 \,\rangle\rangle_t = \begin{cases} z_t^{\varphi_2} \vee (z_t^{\varphi_1} \wedge \langle\langle\, \varphi_1 \,\mathcal{U}\, \varphi_2 \,\rangle\rangle_{t+1}), & t = 1, ..., N-1 \\ z_N^{\varphi_2}. \end{cases}$$

With this definition in place, we define

$$z_t^{\varphi_1 \,\mathcal{U}\, \varphi_2} = z_t^{\varphi_2} \vee (z_t^{\varphi_1} \wedge z_{t+1}^{\varphi_1 \,\mathcal{U}\, \varphi_2})$$

for $t = 1, ..., N-1$, and

$$z_N^{\varphi_1 \,\mathcal{U}\, \varphi_2} = z_N^{\varphi_2} \vee (z_N^{\varphi_1} \wedge (\bigvee_{j=1}^{N} (l_j \wedge \langle\langle\, \varphi_1 \,\mathcal{U}\, \varphi_2 \,\rangle\rangle_j))).$$

Given this encoding of the unbounded until and the encodings of $\square_{[a,b]}$ and $\diamondsuit_{[a,b]}$ above, we can encode

$$z_t^{\varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2} = z_t^{\square_{[0,a]}\, \varphi_1} \wedge z_t^{\diamondsuit_{[a,b]}\, \varphi_2} \wedge z_t^{\diamondsuit_{[a,a]}(\varphi_1 \,\mathcal{U}\, \varphi_2)}.$$

By induction on the structure of STL formulas φ, $z_t^{\varphi} = 1$ if and only if φ holds on the system at time t. With this motivation, given a specification φ, we add a final constraint:

$$z_0^{\varphi} = 1. \tag{2}$$

For a bounded horizon formula, the union of the STL constraints, loop constraints and system constraints gives the MILP encoding of Problem 1; this enables checking feasibility of this set of constraints and finding a solution using an MILP solver. Given an objective function on runs of the system, this approach also enables finding the optimal open-loop trajectory that satisfies the STL specification. Algorithm 1 reviews the procedure for solving Problem 1.

Algorithm 1 Algorithm for Problem 1

procedure OPEN_ LOOP($f, x_0, \mathbf{w}, N, \varphi, J$)
  LOOP_CONSTRAINTS ← Sec. IV-B
  SYSTEM_CONSTRAINTS ← Sec. IV-A
  STL_CONSTRAINTS ← Sec. IV-C2 OR Sec. IV-D

$$\mathbf{u}^* \leftarrow \underset{\mathbf{u} \in \mathcal{U}^N}{\operatorname{argmin}} \quad J(x_0, \mathbf{u}, \mathbf{w}, \varphi)$$
$$\text{s.t.} \quad \text{LOOP\_CONSTRAINTS}$$
$$\text{SYSTEM\_CONSTRAINTS}$$
$$\text{STL\_CONSTRAINTS}$$

  Return $\mathbf{u}^*$
end procedure

D. Quantitative Encoding

The robustness of satisfaction of the STL specification, as defined in II-C, provides a natural objective for the MILP defined in Section IV-C, either in the absence of, or as a complement to domain-specific objectives on runs of the system. The robustness can be computed recursively on the structure of the formula in conjunction with the generation of constraints. 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.

In this section, we sketch the MILP encoding of the predicates and Boolean operators using the quantitative semantics; the encoding of the temporal operators builds on these encodings, as in Section IV-C. 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 $r_t^{\varphi} = \rho^{\varphi}(\mathbf{x}, t)$.

For each predicate $\mu \in P$, we now introduce variables r_t^{μ} for time indices $t = 0, 1, ..., N$, and set $r_t^{\mu} = \mu(x_t)$. To define r_t^{ψ}, where ψ is a Boolean formula, we inductively assume that each operand φ has a corresponding variable $r_t^{\varphi} = \rho^{\varphi}(\mathbf{x}, t)$. Then the Boolean operations are defined as:

Negation: $r_t^{\psi} = \neg r_t^{\phi}$ $\qquad$ $r_t^{\psi} = -r_t^{\phi}$

Conjunction: $r_t^{\psi} = \wedge_{i=1}^{m} r_{t_i}^{\varphi_i}$

$$\sum_{i=1}^{m} p_{t_i}^{\varphi_i} = 1 \tag{3}$$
$$r_t^{\psi} \leq r_{t_i}^{\varphi_i}, i = 1, ..., m \tag{4}$$
$$r_{t_i}^{\varphi_i} - (1 - p_{t_i}^{\varphi_i})M \leq r_t^{\psi} \leq r_{t_i}^{\varphi_i} + M(1 - p_{t_i}^{\varphi_i}) \tag{5}$$

where we introduce new binary variables $p_{t_i}^{\varphi_i}$ for $i = 1, ..., m$, and M is a sufficiently large positive number. Then equation (3) enforces that there is one and only one $j \in \{1, ..., m\}$ such that $b_{t_i}^{\varphi_j} = 1$, equation (4) ensures that r_t^{ψ} is smaller than all $r_{t_i}^{\varphi_i}$, and equation (5) enforces that $r_t^{\psi} = r_{t_j}^{\varphi_j}$ if and only if $b_{t_j}^{\varphi_j} = 1$. Together, these constraints enforce that $r_t^{\psi} = \min_i(r_{t_i}^{\varphi_i})$.

Disjunction: $\psi = \vee_{i=1}^{m} r_{t_i}^{\varphi_i}$ is encoded similarly to conjunction, replacing (4) with $r_t^{\psi} \geq r_{t_i}^{\varphi_i}, i = 1, ..., m$. Using a similar reasoning to that above, this enforces $r_t^{\psi} = \max_i(r_{t_i}^{\varphi_i})$.

The encoding for bounded temporal operators is defined as in Section IV-C; robustness for the unbounded until is defined using $\sup$ and $\inf$ instead of $\max$ and $\min$, but these are equivalent on our finite trajectory representation with discrete time. By induction on the structure of STL formulas φ, this construction yields $r_t^{\varphi} > 0$ if and only if φ is satisfied at time t. Therefore, we can replace the constraints over z_t^{φ} in Section IV-C by these constraints that compute the value of r_t^{φ}, and instead of (2), add the constraint $r_0^{\varphi} > 0$.

Since we consider only the discrete time semantics of STL in this work, the Boolean encoding in Section IV-C could be achieved by converting each formula to LTL, and using existing encodings such as that in [15]. However, the robustness-based encoding we presented in this section has no natural analog for LTL. The advantage of this encoding is that it allows us to maximize the value of r_0^{φ}, obtaining a trajectory that maximizes robustness of satisfaction. Additionally, an encoding based on robustness has the advantage of allowing the STL constraints to be softened or hardened as necessary.

For example, if the original problem is infeasible, we can allow $\rho_0^\varphi > -\epsilon$ for some $\epsilon > 0$, thereby easily modifying the problem to allow a limited violation of the STL property.

The disadvantage is that it is more expensive to compute, due the the additional binary variables introduced during each Boolean operation. Additionally, including robustness as an objective makes the cost function inherently non-convex, with potentially many local minima, and harder to optimize. On the other hand, the robustness constraints are more easily relaxed, allowing us to use a simpler cost function, which can make the problem more tractable.

E. Complexity

In general, our synthesis algorithm has the same complexity as MILPs, which are NP-hard, hence computationally challenging when the dimensions of the problem grow. It is nevertheless appropriate to characterize the computational costs of our encoding and approach in terms of the number of variables and constraints in the resulting MILP. In practice, one measure of problem size is the number of binary variables required to indicate the satisfaction of the predicates μ. This depends directly on the number of predicates used in the STL formula φ.

For the Boolean encoding, if P is the set of predicates used in the formula, then $O(N \cdot |P|)$ binary variables are introduced. In addition, continuous variables are introduced during the MILP encoding of the STL formula. The number of continuous variables used is $O(N \cdot |\varphi|)$, where $|\varphi|$ is the length (i.e. the number of operators) of the formula.

For the robustness-based encoding, $O(N \cdot |P|)$ continuous variables are introduced (one per predicate per time step). In addition, binary variables are introduced during the MILP encoding of each operator in the STL formula. The number of binary variables used is thus $O(N \cdot |\varphi|)$, where $|\varphi|$ is the number of operators of the formula.

Our synthesis algorithm also has polynomial runtime for the following fragment of STL.

Definition 1 (SNN-STL)**.** *Safe Negation-Normal STL (SNN-STL)* is the fragment of STL generated by the recursive grammar

$$\varphi ::= \pi^\mu \mid \neg\pi^\mu \mid \varphi_1 \wedge \varphi_2 \mid \Box_{[a,b]}\ \varphi$$

SNN-STL has the following properties:

- All negations appear only on atomic propositions (pushed down to the leaf nodes of the formula abstract syntax tree).
- The only temporal operators are $\Box$ (with unbounded and bounded intervals).
- Only conjunctions are allowed, no disjunctions.

Such specifications are expressive enough to enforce, e.g., safety specifications in environments where the system state is confined to a conjunction of polyhedra.

Let

$$\texttt{OPEN\_LOOP\_NO\_STL}(f, x_0, N, \varphi, J)$$

denote the procedure that is identical to Algorithm 1, except that the optimization problem in Step 5 is solved with $\texttt{STL\_CONSTRAINTS} = \emptyset$.

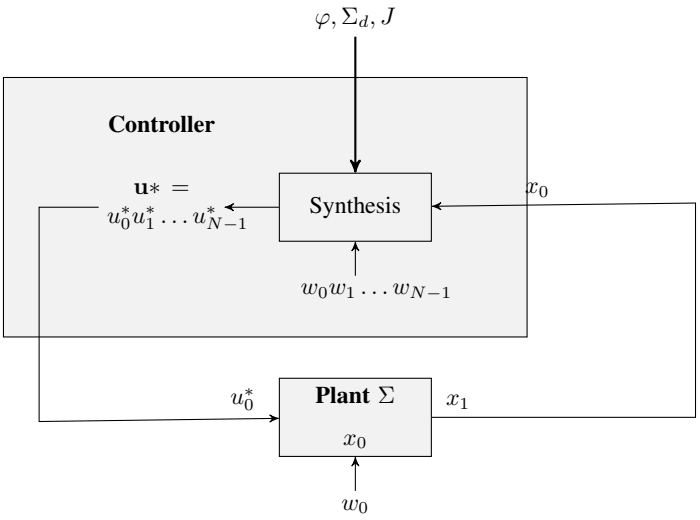

Fig. 2. Closed-loop (MPC) problem formulation. As in the open loop scenario, a sequence of control inputs is synthesized from the specifications, dynamics and cost function. However, at each time step, only the first computed input is used by the plant.

Theorem 1 (Polynomial-time Synthesis for SNN-STL)**.** *Suppose that φ is in SNN-LTL and has linear predicates. Then if* $\texttt{OPEN\_LOOP\_NO\_STL}(f, x_0, \mathbf{w}, N, \varphi, J)$ *is convex, so is* $\texttt{OPEN\_LOOP}(f, x_0, N, \varphi, J)$.

Proof. The proof proceeds by induction on the structure of the formula, showing that the constraints added by `STL_CONSTRAINTS` for each operator restrict the solution to a convex set. First note that since the predicates π^μ are linear, negation of predicates preserves convexity, since $\neg\pi^\mu$ is also linear. Because the intersection of convex sets is convex, the conjunction of a set of convex constraints is also convex. Finally, since the $\Box$ operator is implemented in terms of conjunctions, the constraints imposed by $\Box$ also preserve convexity of the resulting optimization problem. $\Box$

Informally, Theorem 1 states that our encoding of SNN-STL constraints into an MILP preserves convexity in the resulting optimization problem. The resulting optimization problem is therefore encodable as an LP, i.e. without the use of integer variables.

Corollary 1. *Algorithm 1 is polynomial-time for SNN-STL specifications φ.*

V. Model Predictive Control Synthesis

In this section, we will describe a solution to Problem 2 by adding STL constraints to an MPC problem formulation. At each step t of the MPC computation, we will search for a finite trajectory of fixed horizon length H, such that the accumulated trajectory satisfies φ.

A. Synthesis for bounded-time STL formulas

The length of the horizon H is chosen to be at least the bound of formula φ. At time step 0, we will synthesize control $\mathbf{u}^{H,0}$ using the open-loop formulation in Section IV, including

the STL constraints on the length-H trajectory, but without the loop constraints. We will then execute only the first time step $u_0^{H,0}$. At the next step of the MPC, we will solve for $\mathbf{u}^{H,1}$, while constraining the previous values of x_0, u_0 in the MILP, and the STL constraints on the trajectory up to time H. In this manner, we will keep track of the history of states in order to ensure 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.

B. Extension to unbounded formulas

For certain types of unbounded formulas, we can stitch together trajectories of length H using a receding horizon approach, to produce an infinite computation that satisfies the STL formula. An example of this is safety properties, i.e. $\varphi = \Box(\varphi_{MPC})$ for bounded STL formulas φ_{MPC}. For such formulas, at each step of the MPC computation, we will search for a finite trajectory of horizon length H (determined from φ_{MPC} as above) that satisfies φ_{MPC}.

We now describe this approach in more detail. At each step t of the receding horizon control computation, we will employ the open-loop approach in Section IV to find a finite trajectory of fixed horizon length H, such that the trajectory accumulated over time satisfies φ. Given a specification $\varphi = \Box\varphi_{MPC}$, where φ_{MPC} 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 φ at every time step in the trajectory. Note that we omit the loop constraints in this approach, because at each step we search for a finite trajectory, rather than an infinite trajectory with a finite parametrization.

First we define a procedure

$$\texttt{OPEN\_LOOP}^*(f, x_0, \mathbf{w}, N, \Box\varphi_{MPC}, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$$

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 posed in Step 5 is solved without the loop constraints, and with the added constraints:

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

We then define a receding horizon control procedure as in Algorithm 2. At each step, we are optimizing over a horizon of $2H$. We assume available a method $\texttt{PREDICT\_W}(t)$ for predicting the sequence of $2H$ environment inputs starting at time step t.

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_{MPC} 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.

Algorithm 2 MPC Algorithm for Problem 2

1: **procedure** MPC($f, x_0, \phi = \Box\varphi_{MPC}, J$)
2: Let M be a large positive constant.
3: Let H be the bound of φ_{MPC}.
4: Set $P_0 = 0$ and $P_i = -M \ \forall 0 < i \leq H$.
5: $\mathbf{w}^t \leftarrow \texttt{PREDICT\_W}(0)$.
6: Compute $\mathbf{u}^0 = u_0^0, u_1^0, ..., u_{2H-1}^0$ as:

$\mathbf{u}^0 \leftarrow \texttt{OPEN\_LOOP}^*(f, x_0, \mathbf{w}^0, 2H, \Box_{[0,H]}\varphi_{MPC}, J, \mathbf{P}^H, \emptyset)$

7: **for** t=1; t¡=H;t=t+1 **do**
8: Set $\mathbf{u}_{old}^t = u_0^0, u_1^1, u_2^2, ..., u_{t-1}^{t-1}$.
9: Set $P_i = 0$ for $0 \leq i \leq t$, $P_i = -M \ \forall t < i \leq H$.
10: $\mathbf{w}^t \leftarrow \texttt{PREDICT\_W}(t)$.
11: Compute $\mathbf{u}^t = u_0^t, u_1^t, ..., u_{2H-1}^t$ as:

$\mathbf{u}^t \leftarrow \texttt{OPEN\_LOOP}^*(f, x_t, \mathbf{w}^t, 2H, G_{[0,H]}\varphi_{MPC}, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$

12: **end for**
13: **while** `True` **do**
14: Set $\mathbf{u}_{old}^t = u_1^{t-1}, u_2^{t-1}, u_3^{t-1}, ..., u_t^{t-1}$.
15: Set $P_i = 0$ for $0 \leq i \leq H$.
16: $\mathbf{w}^t \leftarrow \texttt{PREDICT\_W}(t)$.

$\mathbf{u}^t \leftarrow \texttt{OPEN\_LOOP}^*(f, x_t, \mathbf{w}^t, 2H, G_{[0,H]}\varphi_{MPC}, J, \mathbf{P}^H, \mathbf{u}_{old}^t)$

17: **end while**
18: **end procedure**

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_{MPC} up to H time steps in the past.

We now show that if Algorithm 2 does not terminate, then the resulting infinite sequence of control inputs enforces satisfaction of the specification $\phi = \Box\varphi_{MPC}$.

Theorem 2. *Let $\phi = \Box\varphi$, and assume that $\mathbf{u}^*$ is an 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 $\mathbf{f}(x_0, \mathbf{u}^*, \mathbf{w}) \models \varphi$.*

Proof. Since H is the bound of φ_{MPC}, the satisfaction of φ_{MPC} 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 $\mathbf{f}(x_t, \mathbf{u}_{old}^{t+H}, \mathbf{w}^H) \models \varphi_{MPC}$. Since $\mathbf{u}^*[t : t+H-1] = \mathbf{u}_{old}^{t+H}$, we see that $\mathbf{f}(x_t, \mathbf{u}^*[t : t+h], \mathbf{w}^H) \models \varphi_{MPC}$.

It follows that $\mathbf{f}(x_0, \mathbf{u}^*, \mathbf{w}) \models \varphi_{MPC}$. □

We have therefore shown how a control input can be

synthesized for infinite sequences satisfying φ, by repeatedly synthesizing control for sequences of length $2H$. A similar approach applies for formulas $\Diamond \varphi_{MPC}$ and $\varphi_{MPC}\, \mathcal{U}\, \psi_{MPC}$, where $\varphi_{MPC}, \psi_{MPC}$ are bounded-time.

Note that we assumed that PREDICT_W(t) returns an exact prediction of the disturbance signal over the next $2H$ time steps. The correctness of our approach relies on this assumption. An interesting direction of future work is to relax this requirement, demanding only an uncertain prediction of the disturbance signal.

Remark 2. The control objective for MPC is usually to steer the state to the origin or to an equilibrium state. Questions that arise include those of ensuring feasibility at each time step, closed-loop stability and near-optimal performance [19]. There is a mature theory of stability for MPC, where the essential ingredients are terminal costs, terminal constraint sets, and local stabilizing controller that ensure closed-loop stability [5].

In this work, our control objective is not closed-loop stability, but satisfaction of an STL formula. We achieve this, as detailed above, through choice of a sufficiently large prediction horizon H. This can be compared with the manner in which automatic satisfaction of a terminal constraint is sometimes attained by prior choice of a sufficiently large horizon.

VI. Experimental Comparison of Encodings

We implemented the Boolean and robust encodings using the tools Breach [20] and Yalmip [21], and now present results obtained with the following formulas:

- $\varphi_1 = \Box_{[0,0.1]}\, x_t^{(1)} > 0.1$
- $\varphi_2 = \Box_{[0,0.1]}(x_t^{(1)} > 0.1) \wedge \Box_{[0,0.1]}(x_t^{(2)} < -0.5)$
- $\varphi_3 = \Box_{[0,0.5]} \Diamond_{[0,0.1]}(x_t^{(1)} > 0.1)$
- $\varphi_4 = \Diamond_{[0,0.2]}(x_t^{(1)} > 0.1 \wedge (\Diamond_{[0,0.1]}(x_t^{(2)} > 0.1) \wedge \Diamond_{[0,0.1]}(x_t^{(3)} > 0.1)))$

In this study, we used the trivial system $\mathbf{x} = \mathbf{u}$, where $\mathbf{x}$ is a 3-dimensional signal (i.e. $x_t = x_t^{(1)} x_t^{(2)} x_t^{(3)}$), so that no constraint is generated for the system dynamics, and the cost function $J(\mathbf{x}, \mathbf{u}) = \sum_{k=1}^{N} \|\mathbf{u}_{t_k}\|_1$. Note that the output of this procedure for a formula φ is a signal of minimal norm which satisfies φ when using the Boolean encoding and which satisfies φ with a specified robustness $\rho^{\varphi}(\mathbf{x}) = 0.1$ for the robust encoding. For each formula we computed the Boolean and robust encodings for an horizon $N = 30$ and sampling time $\tau = 0.025s$ and report the number of constraints generated by each encoding, the time to create the resulting MILP with Yalmip and the time to solve it using the solver Gurobi.[1] All experiments were run on a laptop with an Intel Core i7 2.3 GHz processor and 16 GB of memory.

A first observation is that for both encodings, most of the time is spent creating the MILP, while solving it is done in a fraction of a second. Also, while the robust encoding generates 3 to 5 times more constraints, the computational time to create and solve the corresponding MILPs is hardly twice more. The exception is solving the MILP for φ_4, which takes significantly more time for the robust encoding than for the Boolean encoding. The reason is hard to pinpoint without a more thorough investigation, but we can already note that solving a MILP is NP-hard, and while solvers use sophisticated heuristics to mitigate this complexity, instances for which these heuristics fail are bound to appear.

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

Formula	#constraints		Yalmip Time (s)		Solver time (s)	
	B	R	B	R	B	R
φ_1	154	488	1.71	2.04	0.0070	0.0085
φ_2	364	897	1.94	2.69	0.0115	0.0229
φ_3	244	1282	1.84	3.15	0.0064	0.1356
φ_4	574	1330	2.29	3.37	0.2167	238.6

TABLE I
Boolean (B) vs robust (R) encodings. Yalmip time represents the time taken by the tool Yalmip in order to generate the MILP and Solver time is the time taken by the solver Gurobi to actually solve it.

VII. Case Study: Building Climate Control

A. Mathematical Model of a Building

Next we consider the problem of controlling building indoor climate, using the model proposed by Maasoumy et al [22]. In this section we present a summary of the building's thermal model.

As shown in Fig. 3, the building is modeled as a resistor-capacitor circuit with n nodes, m of which are rooms and the remaining $n-m$ are walls. We denote the temperature of room r_i by T_{r_i}. The wall and temperature of the wall between rooms i and j are denoted by $w_{i,j}$ and $T_{w_{i,j}}$, respectively. The temperature of wall $w_{i,j}$ and room r_i are governed by the following equations:

$$C_{i,j}^{w} \frac{dT_{w_{i,j}}}{dt} = \sum_{k \in \mathcal{N}_{w_{i,j}}} \frac{T_{r_k} - T_{w_{i,j}}}{R_{i,j_k}} + r_{i,j} \alpha_{i,j} A_{w_{i,j}} Q_{rad_{i,j}} \tag{6}$$

$$C_i^{r} \frac{dT_{r_i}}{dt} = \sum_{k \in \mathcal{N}_{r_i}} \frac{T_k - T_{r_i}}{R_{i,k_i}} + \dot{m}_{r_i} c_a (T_{s_i} - T_{r_i}) + w_i \tau_{w_i} A_{win_i} Q_{rad_i} + \dot{Q}_{int_i}, \tag{7}$$

where $C_{i,j}^{w}$, $\alpha_{i,j}$ and $A_{w_{i,j}}$ are heat capacity, a radiative heat absorption coefficient, and the area of $w_{i,j}$, respectively. R_{i,j_k} is the total thermal resistance between the centerline of wall (i,j) and the side of the wall on which node k is located. $Q_{rad_{i,j}}$ is the radiative heat flux density on $w_{i,j}$. $\mathcal{N}_{w_{i,j}}$ is the set of all neighboring nodes to $w_{i,j}$. $r_{i,j}$ is a wall identifier, which equals 0 for internal walls and 1 for peripheral walls, where either i or j is the outside node. T_{r_i}, C_i^{r} and $\dot{m}_{r_i}$ are the temperature, heat capacity and air mass flow into room i, respectively. c_a is the specific heat capacity of air, and T_{s_i} is the temperature of the supply air to room i. w_i is a window identifier, which equals 0 if none of the walls surrounding room i have windows, and 1 if at least one of them does. τ_{w_i} is the transmissivity of the glass of window i, A_{win_i} is the total area of the windows on walls surrounding room i, Q_{rad_i} is the radiative heat flux density per unit area radiated to room i, and $\dot{Q}_{int_i}$ is the internal heat generation in room i. $\mathcal{N}_{r_i}$ is

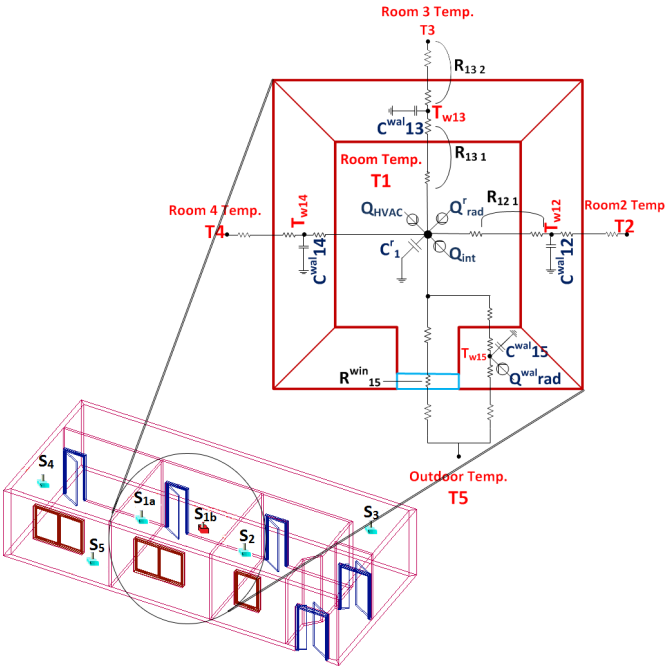

Fig. 3. Resistor-capacitor representation of a typical room with a window.

the set of neighboring *room* nodes for room i. Further details on this thermal model can be found in [22].

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

$$\dot{x}_t = f(x_t, u_t, w_t).$$

Here $x_t \in \mathbb{R}^n$ is the state vector representing the temperature of the nodes in the thermal network, 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. two for air mass flow and supply air temperature). The HVAC system of the building considered for this study operates with a constant supply air temperature, while air mass flow is the time varying control input. Hence, in the following simulations we consider supply air temperature constant and treat air mass flow as the control signal. Vector w_t stores the estimated disturbance values, aggregating various unmodelled dynamics such as T_{out}, $\dot{Q}_{int}$ and Q_{rad}, and can be estimated using historical data [23]. $y_t \in \mathbb{R}^m$ is the output vector, representing the temperature of the thermal zones. The building model was trained using historical data, and the result of the system identification is shown in Fig. 4.

B. MPC for Building Climate Control

We consider the problem of controlling the above building's HVAC system using an MPC scheme. We adopt the MPC formulation proposed by Maasoumy et al. [24], with the objective of minimizing the total energy cost (in dollar value). τ and H denote the length of each time slot and the prediction horizon (in number of time slots) of the MPC, respectively. Assume that the system dynamics are also discretized with

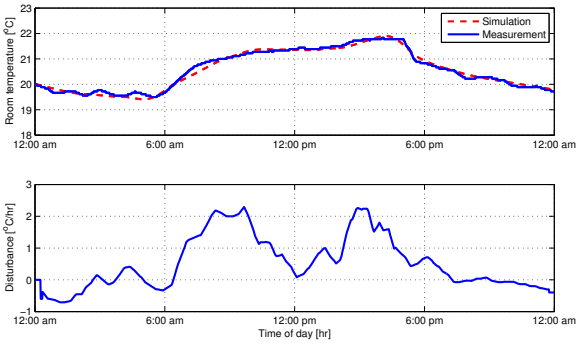

Fig. 4. Simulated temperature, measured temperature and unmodelled dynamics of a thermal zone in Bancroft library on UC Berkeley campus.

a sampling time of τ. Here we consider $\tau = 0.5$ hr and $H = 24$. At each time t, the predictive controller solves an optimal control problem to compute $\vec{u}_t = [u_t, \ldots, u_{t+H-1}]$, and minimizes 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 MPC is to maintain a comfort temperature given by T^{comf} whenever the room is occupied while minimizing the cost of heating. This problem can be expressed as follows:

$$\begin{aligned}
&\min_{\vec{u}_t} \sum_{k=0}^{H-1} \|u_{t+k}\| \quad \text{s.t.} \\
&x_{t+k+1} = f(x_{t+k}, u_{t+k}, w_{t+k}), \\
&x_t \models \varphi \quad \text{with} \quad \varphi = \square_{[0,H]}((\text{occ}_t > 0) \Rightarrow (T_t > T_t^{\text{comf}})) \\
&u_{t+k} \in \mathcal{U}_{t+k}, \ \ k = 0, ..., H-1
\end{aligned}$$

The STL formula was encoded using the robust MILP encoding and results are presented in Fig. 5. Again we observed that creating the MILP structure was longer than solving an instance of it (4.1s versus 0.15s). However, by using a proper parametrization of the problem in Yalmip, the creation of the MILP structure can be done once offline and reused online for each step of the MPC, which makes the approach promising and potentially applicable even for real-time applications.

VIII. Case Study II: Regulation Control for Smart Grid

A. Mathematical Model

The second case study we consider is the n-areas smart grid model presented in [16] and depicted in Fig. 6. The interconnection of power system components, including a governor, turbine and generator in each area is shown in the block diagram in Fig. 7. In the diagram, δP_C is a control input which acts against an increase or decrease in power demand to regulate the system frequency ω, and δP_D denotes fluctuations in power demand, modeled as an exogenous input (disturbance). Under steady state, we have: $\omega = \omega_o$ and $P_M = P_G = P_M^o$, where ω_o, V_t^o, and P_M^o are the nominal values for rated frequency, terminal voltage and mechanical power input.

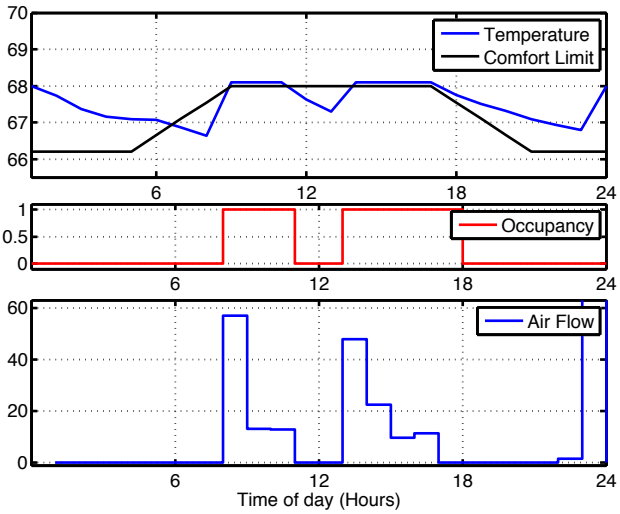

Fig. 5. Room temperature control with constraints based on occupancy, expressed in STL.

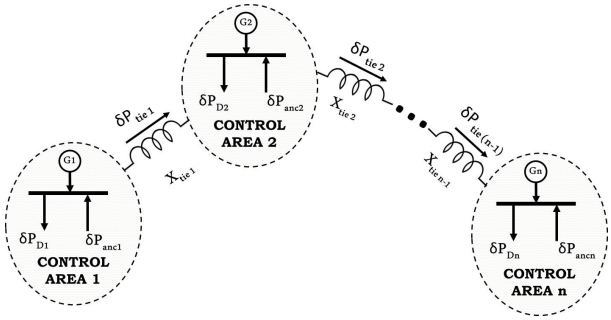

Fig. 6. Power system grid with n areas. The dynamics in each area is depicted in Fig.7.

Next, we present the mathematical model for one area i (note that superscripts refer to the control area, and subscripts index states in each area).

$$\frac{dx_1^i}{dt} = \frac{(-D^i x_1^i + \delta P_M^i - \delta P_D^i - \delta P_{\text{tie}}^i + \delta P_{\text{anc}}^i)}{M}, \tag{8a}$$

$$\frac{dx_2^i}{dt} = \frac{(x_3^i - x_2^i)}{T_7^i}, \quad \frac{dx_3^i}{dt} = \frac{(x_4^i - x_3^i)}{T_6^i}, \quad \frac{dx_4^i}{dt} = \frac{(x_5^i - x_4^i)}{T_5^i}, \tag{8b}$$

$$\frac{dx_5^i}{dt} = \frac{(P_{GV}^i - x_5^i)}{T_4^i}, \quad \frac{dx_6^i}{dt} = \frac{(x_7^i - x_6^i)}{T_3^i}, \tag{8c}$$

$$\frac{dx_7^i}{dt} = \frac{(-x_7^i + \delta P_C^i - x_1^i/R^i)}{T_1^i}, \quad \frac{dx_8^i}{dt} = x_1^i \tag{8d}$$

where δP_M^i and P_{GV}^i are given by $\delta P_M^i = K_1^i x_5^i + K_3^i x_4^i + K_5^i x_3^i + K_7^i x_2^i$, and $P_{GV}^i = (1 - T_2/T_3)x_6^i + (T_2/T_3)x_7^i$. D is the damping coefficient, M is the machine inertia constant, R is the speed regulation constant, T_i's are time constants for power system components, and K_i's are fractions of total mechanical power outputs associated with different operating parts of the turbine. δP_{tie}^i represents power transfer from area i to other areas. In equation (8), the first state represents the frequency increment, $x_1^i = \delta\omega_i$. It can be shown that P_{tie}^i can be obtained from

$$\delta P_{\text{tie}}^{ij} = \sum_{j=1}^{n} \nu_{ij}(x_8^i - x_8^j), \tag{9}$$

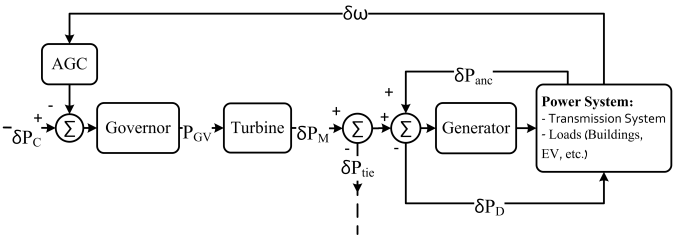

Fig. 7. Block diagram of power system and its relation to governor, turbine, generator, and the AGC signal for each control area. More details on the power grid model can be found in [16].

where ν_{ij} is the transmission line stiffness coefficient, and the state variable x_8^i is the integral of x_1^i.

The classical AGC implements a simple PI control to regulate the grid frequency. In a multi-area power system, in addition to regulating frequency within each area, the auxiliary control should maintain the net interchange power with neighboring areas at scheduled values [25]. This is generally accomplished by adding a tie-line flow deviation to the frequency deviation in the auxiliary feedback control loop. A suitable linear combination of the frequency and tie-line deviations for area i, is known as the Area Control Error (ACE): this measures the difference between the scheduled and actual electrical generation within a control area while taking frequency bias into account. The ACE of area i is thus defined as $ACE^i = \delta P_{tie}^i + \beta^i x_1^i$, and β^i is the bias coefficient of area i. The standard industry practice is to set the bias β^i at the so-called Area Frequency Response Characteristic (AFRC), which is defined as $\beta^i = D^i + 1/R^i$. The integral of ACE is used to construct the speed changer position feedback control signal (δP_C^i), i.e., $\delta P_C^i = -K^i x_9^i$, where K^i is the feedback gain and $\frac{dx_9^i}{dt} = ACE^i$.

The resulting state space model can be discretized and written in compact form as

$$x(t_{k+1}) = Ax(t_k) + B_1 u_{\text{anc}}(t_k) + B_2 w(t_k). \tag{10}$$

Where $u_{\text{anc}} = [\delta P_{\text{anc}}^1 \dots \delta P_{\text{anc}}^n]^T$ are the ancillary inputs, and the exogenous inputs (i.e. disturbances or variations in demands) are denoted by $w = [\delta P_D^1 \dots \delta P_D^n]^T$. We propose controller synthesis for the ancillary services, complementing the primary control of AGC.

B. MPC for Ancillary Services

We require that u_{anc} be bounded and satisfies a maximum ramp constraint, i.e., $\underline{u}_{\text{anc}} \le u_{\text{anc}}(t_k) \le \overline{u}_{\text{anc}}$ with $\underline{u}_{\text{anc}} > 0$ and $|u_{\text{anc}}(t_{k+1}) - u_{\text{anc}}(t_k)| \le \lambda$, for some $\lambda > 0$. At each time step k, we thus solve the following problem:

$$\begin{aligned} \min_{U_{\text{anc}}(k)} \quad & J(\text{ACE}, U_{\text{anc}}) \\ \text{s.t.} \quad & x(t_{k+1}) = Ax(t_k) + B_1 u_{\text{anc}}(t_k) + B_2 w(t_k) \\ & \underline{u}_{\text{anc}} \le u_{\text{anc}}(t_{k+j}) \le \overline{u}_{\text{anc}} \\ & |u_{\text{anc}}(t_{k+j+1}) - u_{\text{anc}}(t_{k+j})| \le \lambda \end{aligned} \tag{11}$$

where $U_{\text{anc}}(k) = (u_{\text{anc}}(t_k), u_{\text{anc}}(t_{k+1}), \ \dots, u_{\text{anc}}(t_{k+H}))$ is the vector of inputs from k to $k+H$ and H is the prediction

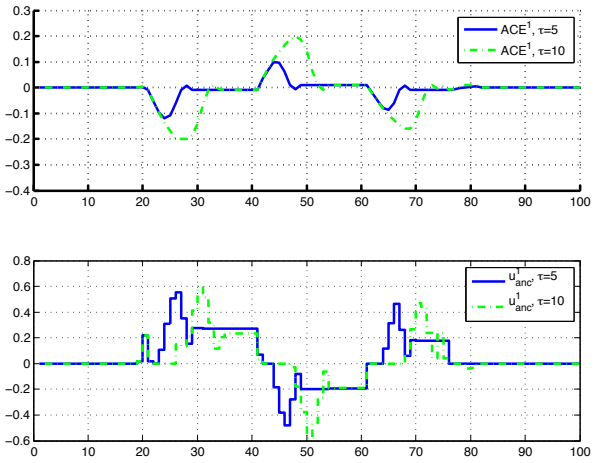

Fig. 8. Comparison of ACE^1 (Area Control Error of Control Area 1) stabilization using φ_t in (12) with $\tau = 5$ and $\tau = 10$. The controller enforces the stabilization delay in both cases.

horizon. All the constraints of problem (11) that depend on j should hold for $j = 0, 1, \ldots, H-1$.

The cost function proposed in [16] minimizes the ℓ_2 norm of the ACE signal in areas $i = 1, \ldots, n$, by exploiting the ancillary service available in each area, while taking into account the system dynamics and constraints. We propose to constrain the ACE signal to satisfy a specified set of STL properties, while minimizing the ancillary service used by each area. Thus we defined $J(ACE, U_{\mathrm{anc}}) = \|U_{\mathrm{anc}}\|_{\ell_2} = \sum_{i=1}^{2}\sum_{j=0}^{H-1}(U^i_{\mathrm{anc}}[k+j])^2$, and an STL formula φ which says that whenever $|\mathrm{ACE}^i|$ is larger than 0.01, it should become less than 0.01 in less than τ s. More precisely we used $\varphi = \Box(\varphi_t)$ with

$$\begin{aligned}\varphi_t = \neg(|\mathrm{ACE}^1| < .01)) \Rightarrow (\Diamond_{[0,\tau]}(|\mathrm{ACE}^1| < .01) \\ \wedge (\neg(|\mathrm{ACE}^2| < .01)) \Rightarrow (\Diamond_{[0,\tau]}(|\mathrm{ACE}^2| < .01)\end{aligned} \quad (12)$$

We encoded this formula and added the resulting constraints to the MPC problem as described in the previous sections, and solved it for different values of τ. Results are shown in Fig. 8, and demonstrate that the STL constraint is correctly enforced in the stabilization of the ACE signal.

IX. Related Work

Receding horizon control for temporal logic has been considered before in the context of LTL [26], where the authors propose a reactive synthesis scheme for specifications with GR(1) goals. The authors in [27] also propose an MPC scheme for specifications in synthetically co-safe LTL – our approach extends synthesis capabilities to a wider class of temporal logic specifications. In [28], the authors consider full LTL but use an automata-based approach, involving potentially expensive computations of a finite state abstraction of the system and a Buchi automaton for the specification. We circumvent these expensive operations using a BMC approach to synthesis. In [17], the authors present a model predictive control scheme to stabilize mixed logical dynamical systems on desired reference trajectories, while fulfilling propositional logic constraints and heuristic rules. A major contribution of this work is to extend the constraint specification language for such systems to STL specifications, which allow expression of complex temporal properties including safety, liveness, and response.

Our work extends the standard BMC paradigm for finite discrete systems [14] to STL, which accommodates continuous systems. In BMC, discrete state sequences of a fixed length, representing counterexamples or plans, are obtained as satisfying assignments to a Boolean satisfiability (SAT) problem. The approach has been extended to hybrid systems, either by computing a discrete abstraction of the system [29], [30] or by extending SAT solvers to reason about linear inequalities [31], [32]. Similarly, MILP encodings inspired by BMC have been used to generate trajectories for continuous systems with Linear Temporal Logic specifications [15], [33], [34], and for a restricted fragment of Metric Temporal Logic without nested operators [35]. However, this is the first work to consider a BMC approach to synthesis for full STL.

X. Concluding Remarks

The main contribution of this paper is a pair of bounded model checking style encodings for signal temporal logic specifications as mixed integer linear constraints. We showed how our encodings can be used to generate control for systems that must satisfy STL properties, and additionally to ensure maximum robustness of satisfaction. Our formulation of the STL synthesis problem can be used as part of existing controller synthesis frameworks to compute feasible and optimal controllers for cyber-physical systems. We presented experimental results for controller synthesis on simplified models of a smart micro-grid and HVAC system, and showed how the MPC schemes in these examples can be framed in terms of synthesis from an STL specification, with simulation results illustrating the effectiveness of our proposed synthesis.

We have demonstrated the ability to synthesize control for systems on both the demand and supply sides of a smart grid. We view this as progress toward a contract-based framework for specifying and designing components of the smart grid and their interactions using STL specifications. Future work includes a *reactive synthesis* approach to synthesizing control inputs for systems operating in uncertain environments: we have already demonstrated preliminary results in this direction in [36]. We will also further explore synthesis in an MPC framework for unbounded STL properties. As mentioned in Section V-B, this is an easy extension of our approach for certain types of properties. Extending this to arbitrary properties has ties to online monitoring of STL properties [37], which is another direction of further exploration.

References

[1] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343 – 352, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510980800455X

[2] 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*, vol. PP, no. 99, pp. 1–1, 2013.

[3] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[4] 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*. Wiley-Interscience, 2002, pp. 149–174.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. [Online]. Available: http://dx.doi.org/10.1016/S0005-1098(99)00214-9

[6] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FORMATS/FTRTFT*, 2004, pp. 152–166.

[7] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2009.06.021

[8] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *FORMATS*, 2010, pp. 92–106.

[9] M. Maasoumy, P. Nuzzo, F. Iandola, M. Kamgarpour, A. Sangiovanni-Vincentelli, and C. Tomlin, "Optimal load management system for aircraft electric power distribution," in *IEEE Conference on Decision and Control (CDC)*, 2013.

[10] A. Biere, K. Heljanko, T. A. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Logical Methods in Computer Science*, vol. 2, no. 5, 2006.

[11] V. Raman, M. Maasoumy, and A. Donzé, "Model predictive control from signal temporal logic specifications: A case study," in *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, ser. CyPhy'14. New York, NY, USA: ACM, 2014, pp. 52–55. [Online]. Available: http://doi.acm.org/10.1145/2593458.2593472

[12] 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.

[13] G. E. Fainekos and G. J. Pappas, "Robust sampling for MITL specifications," in *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, 2007, pp. 147–162. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75454-1_12

[14] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *TACAS*, 1999, pp. 193–207.

[15] 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*, 2014, pp. 5319–5325. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6907641

[16] M. Maasoumy, B. M. Sanandaji, A. Sangiovanni-Vincentelli, and K. Poolla, "Model predictive control of regulation services from commercial buildings to the smart grid," in *IEEE American Control Conference (ACC)*, 2014.

[17] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999. [Online]. Available: http://dx.doi.org/10.1016/S0005-1098(98)00178-2

[18] A. Donzé, T. Ferrère, and O. Maler, "Efficient robust monitoring for stl," in *CAV*, 2013, pp. 264–279.

[19] M. Morari and J. Lee, "Model predictive control: Past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999. [Online]. Available: https://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=1641

[20] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *CAV*, 2010, pp. 167–170.

[21] J. Lfberg, "Yalmip : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[22] M. Maasoumy Haghighi, "Controlling energy-efficient buildings in the context of smart grid: A cyber physical system approach," Ph.D. dissertation, University of California, Berkeley, Dec 2013. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-244.html

[23] M. Maasoumy and A. Sangiovanni-Vincentelli, "Total and peak energy consumption minimization of building hvac systems using model predictive control," *Design Test of Computers, IEEE*, vol. 29, no. 4, pp. 26 –35, aug. 2012.

[24] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. Sangiovanni-Vincentelli, "Selecting building predictive control based on model uncertainty," in *IEEE American Control Conference (ACC 2014)*, Portland, USA, June 2014.

[25] H. Bevrani, *Robust Power System Frequency Control*, ser. Power Electronics and Power Systems. Springer, 2009.

[26] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Automat. Contr.*, vol. 57, no. 11, pp. 2817–2830, 2012.

[27] 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*, 2013, pp. 343–352. [Online]. Available: http://doi.acm.org/10.1145/2461328.2461379

[28] X. C. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2013.11.030

[29] G. J. P. Nicoló Giorgetti and A. Bemporad, "Bounded model checking of hybrid dynamical systems," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec 2005, pp. 672–677.

[30] S. Jha, B. A. Brady, and S. A. Seshia, "Symbolic reachability analysis of lazy linear hybrid automata," in *Proc. 5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, ser. Lecture Notes in Computer Science, vol. 4763, 2007, pp. 241–256.

[31] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani, "Verifying industrial hybrid systems with MathSAT," *Electronic Notes in Theoretical Computer Science*, vol. 119, no. 2, pp. 17 – 32, 2005, proceedings of the 2nd International Workshop on Bounded Model Checking (BMC 2004) Bounded Model Checking 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105000885

[32] M. Franzle and C. Herde, "Efficient proof engines for bounded model checking of hybrid systems," *Electronic Notes in Theoretical Computer Science*, vol. 133, no. 0, pp. 119 – 137, 2005, proceedings of the Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2004) Formal Methods for Industrial Critical Systems 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105050279

[33] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, Dec 2008, pp. 2117–2122.

[34] Y. Kwon and G. Agha, "Ltlc: Linear temporal logic for control," in *HSCC*, M. Egerstedt and B. Mishra, Eds., 2008, pp. 316–329.

[35] 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*, 2008, pp. 3953–3958. [Online]. Available: http://dx.doi.org/10.1109/CDC.2008.4739366

[36] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Hybrid Systems: Computation and Control, HSCC 2015, Seattle, WA, USA, April 14-16, 2015*, 2015, pp. 239–248.

[37] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," in *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, 2015, pp. 55–70. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23820-3_4