Model Predictive Control with Signal Temporal Logic Specifications

Vasumathi Raman[1], Alexandre Donzé[2], Mehdi Maasoumy[2],
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 discrete-time 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 model predictive control framework. We present experimental results for controller synthesis for building energy and climate control.**

I. Introduction

Temporal logics provide a compact mathematical formalism for specifying desired behaviors of a system. In particular, algorithms for verification and synthesis for these logics enable construction of discrete supervisory controllers satisfying the specified properties. These discrete controllers have successfully been used to construct hybrid controllers for cyber-physical systems in domains including robotics [9] and aircraft power system design [25]. 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 of such systems is in general undecidable [1].

Signal Temporal Logic (STL) [22] was originally developed in order to specify and monitor the expected behavior of physical systems, including temporal constraints between events. 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 the circuit. STL has the advantage of naturally admitting a *quantitative* semantics which, in addition to the yes/no answer to the satisfaction question, provides a real number grading the quality of the satisfaction or violation. Such semantics have been defined for timed logics including Metric Temporal Logic (MTL) [10] and STL [8] to assess the *robustness* of the systems to parameter or timing variations.

Model Predictive Control (MPC) is based on iterative, finite horizon, discrete time optimization of a model of the plant. 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 $t+H$. Only the first step of the computed optimal control strategy is implemented; the plant state is then sampled again, and new calculations are performed on a horizon of H starting from the new current state. While the global optimality of this sort of "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 [23].

In this paper, we frame MPC in terms of control synthesis from STL specifications. The objective is to 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. Recent work on optimal control synthesis of aircraft load management systems [18] 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 STL specifications.

Receding horizon control for temporal logic has been considered before in the context of Linear Temporal Logic (LTL) [28], where the authors propose a reactive synthesis scheme for specifications with GR(1) goals. The authors in [12] also propose an MPC scheme for specifications in syntactically co-safe LTL – our approach extends synthesis capabilities to a wider class of temporal logic specifications. In [5], 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 Bounded Model Checking (BMC) approach to synthesis. In [3], the authors present a 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 was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

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

A. Donzé, M. Maasoumy, 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, maasoumy@eecs.berkeley.edu, alberto@berkeley.edu, sseshia@eecs.berkeley.edu

this work is to extend the constraint specification language for such systems to STL.

Our work extends the standard BMC paradigm for finite discrete systems [4] 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 [24], [13] or by extending SAT solvers to reason about linear inequalities [2], [11]. Similarly, MILP encodings inspired by BMC have been used to generate trajectories for continuous systems with Linear Temporal Logic specifications [15], [16], [27], and for a restricted fragment of Metric Temporal Logic without nested operators [14]. However, this is the first work to consider a BMC approach to synthesis for full STL.

Our main contribution is a pair of BMC-style encodings 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 also demonstrate how our MILP formulation of the STL synthesis problem can be used as part of existing MPC frameworks, to compute feasible and optimal controllers for cyber-physical systems under timed specifications. We present experimental results comparing both encodings, and a case study of controller synthesis on a model of a Heating Ventilation and Air Conditioning (HVAC) system; another case study on the control of a smart-building level microgrid was previously reported in a work-in-progress version of this paper [26]. 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. Discrete-Time Continuous Systems

We consider discrete-time continuous systems of the form

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

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, and $x_0 \in \mathcal{X}$ is the initial state. A *run* $\mathbf{x} = x_0 x_1 x_2 ...$ is an infinite sequence of its states, where $x_t \in \mathcal{X}$ is the state of the system at index t, and for each $t = 0, 1, ...$, there exists a control input $u_t \in U$ such that $x_{t+1} = f(x_t, u_t)$. Given an initial state x_0 and a control input sequence $\mathbf{u}^N = u_0 u_1 u_2 \ldots u_{N-1}$, the resulting horizon-$N$ run of a system modeled by (1), which we denote by $\mathbf{x}(x_0, \mathbf{u}^N) = x_0 x_1 x_2 ... x_N$ is unique. Note that a horizon-N run as defined here has $N+1$ states. We also introduce a generic cost function $J(\mathbf{x}, \mathbf{u})$ that maps runs to $\mathbb{R}$.

B. Signal Temporal Logic

We assume that STL formulas are provided in negation normal form, so all negations appear in front of predicates. An STL formula can always be rewritten as a negation normal form formula of length linear in the original length. STL formulas are thus defined recursively as:

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

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:

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

A signal $\mathbf{x} = x_0 x_1 x_2 ...$ satisfies φ, denoted by $\mathbf{x} \models \varphi$, if $(\mathbf{x}, 0) \models \varphi$. Informally, $\mathbf{x} \models \Box_{[a,b]}\, \varphi$ if the property defined by 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.

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

C. 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:

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

III. Problem Statement

We now formally state the STL control synthesis problem and its MPC formulation.

Problem 1 (Optimal Controller Synthesis from STL): Given a system of the form (1), initial state x_0, trajectory length N, STL formula φ and cost function J, compute

$$\begin{array}{ll}
\text{argmin}_{\mathbf{u}^N} & J(\mathbf{x}(x_0,\mathbf{u}^N)) \\
 & \text{s.t. } \mathbf{x}(x_0,\mathbf{u}^N)\models\varphi
\end{array}$$

Problem 2 (MPC from STL Specifications): Given a system of the form (1), initial state x_0, STL formula φ and cost function J, at each time step t, compute

$$\begin{array}{ll}
\text{argmin}_{\mathbf{u}^{H,t}} & J(\mathbf{x}(x_t,\mathbf{u}^{H,t}),\mathbf{u}^{H,t})) \\
 & \text{s.t. } \mathbf{x}(x_0,\mathbf{u})\models\varphi,
\end{array}$$

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 t, and $\mathbf{u} = u_0^{H,0}u_0^{H,1}u_0^{H,2}....$

In Sections IV and VI, we present both an open-loop solution to Problem 1, and a solution to Problem 2 for bounded-time 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

The encoding of Problem 1 as an MILP consists of system 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. 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 $\mathbf{x}(x_0,\mathbf{u}_N)$ 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. Examples include a building-level smart grid control system model [20] and mixed-logical dynamical systems such as those presented by Bemporad and Morari [3].

B. 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 $\mu(x_t) > 0$:

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

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. Note that $z_t = 1$ if and only if $\mu(x_t) > 0$. 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-B.1, 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 if not. Here we define Boolean operations on these variables; these are the building blocks of our recursive encoding. The definitions in this subsection are identical to those in [27].

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

$$\begin{array}{ll}
\underline{\text{Negation: } z_t^{\psi} = \neg z_t^{\varphi}} & z_t^{\psi} = 1 - z_t^{\varphi} \\
\underline{\text{Conjunction: } z_t^{\psi} = \wedge_{i=1}^m z_{t_i}^{\varphi_i}} & \begin{array}{l} z_t^{\psi} \leq z_{t_i}^{\varphi_i}, i = 1,...,m, \\ z_t^{\psi} \geq 1 - m + \sum_{i=1}^m z_{t_i}^{\varphi_i} \end{array} \\
\underline{\text{Disjunction: } z_t^{\psi} = \vee_{i=1}^m z_{t_i}^{\varphi_i}} & \begin{array}{l} z_t^{\psi} \geq z_{t_i}^{\varphi_i}, i = 1,...,m, \\ z_t^{\psi} \leq \sum_{i=1}^m z_{t_i}^{\varphi_i} \end{array}
\end{array}$$

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.

$$\begin{array}{l}
\underline{\text{Always: } \psi = \Box_{[a,b]}\,\varphi} \\
\text{Let } a_t^N = \min(t+a,N) \text{ and } b_t^N = \min(t+b,N) \\
\text{Define } z_t^{\psi} = \wedge_{i=a_t^N}^{b_t^N} z_i^{\varphi}
\end{array}$$

The logical operation $\wedge$ on the variables z_i^{φ} here is as defined in Section IV-B.2. 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 $z_t^\psi = \vee_{i=a_t^N}^{b_t^N} z_i^\varphi$ 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 [7]:

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

We 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

$$z_N^{\varphi_1 \, \mathcal{U} \, \varphi_2} = z_N^{\varphi_2}.$$

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

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

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:

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

The union of the STL constraints and system constraints gives the MILP encoding of Problem 1; this enables checking feasibility of this set 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.

Mixed integer-linear programs are NP-hard, hence computationally challenging when the dimensions of the problem grow. Hence, the computational costs of our encoding and approach are in terms of the number of variables and constraints in the resulting MILP. 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.

V. Robustness-Based 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-B.3, 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 encoding of the predicates and Boolean operators using an MILP; the encoding of the temporal operators builds on these encodings as in Section IV-B.3. 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 $r_t^\varphi = \rho^\varphi(\mathbf{x}, t)$.

For each predicate $\mu \in P$, we now introduce variables r_t^μ for time indices $t = 0, 1, ..., N$, and set $r_t^\mu = \mu(x_t)$. For r_t^ψ where ψ is a Boolean formula, we 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 (3) enforces that there is one and only one $j \in \{1, ..., m\}$ such that $p_{t_i}^{\varphi_j} = 1$, (4) ensures that r_t^ψ is smaller than all $r_{t_i}^{\varphi_i}$, and (5) enforces that $r_t^\psi = r_{t_j}^{\varphi_j}$ if and only if $p_{t_j}^{\varphi_j} = 1$. 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-B.3; 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-B.3 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-B.3 could be achieved by converting each formula to LTL, and using existing encodings such as that in [27]. 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 minimums, and harder to optimize. On the other hand, the robustness constraints are more easily relaxed, which allows us to use a simpler cost function, which can make the problem more tractable.

VI. 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. 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 in Section VI-A) that satisfies φ_{MPC}.

VII. Experimental Results

A. Boolean vs Robust Encoding

We implemented the Boolean and robust encodings on top of the tools Breach [6] and Yalmip [17] and first present preliminary experimental 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 2.3GHz processor and 16GB of memory.

Formula	#constraints		Yalmip Time (s)		Solver time (s)	
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 (left) vs robust (right) 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.

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.

B. Mathematical Model of a Building

Next we consider the problem of controlling building indoor climate, using the model proposed by Maasoumy et al [21].

1) Heat Transfer: As shown in Fig. 1, a building is modeled as a resistor-capacitor circuit with n nodes, m of which are rooms and the remaining $n-m$ 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 $w_{i,j}$ and r_i room are governed by the following equation:

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

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

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.

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

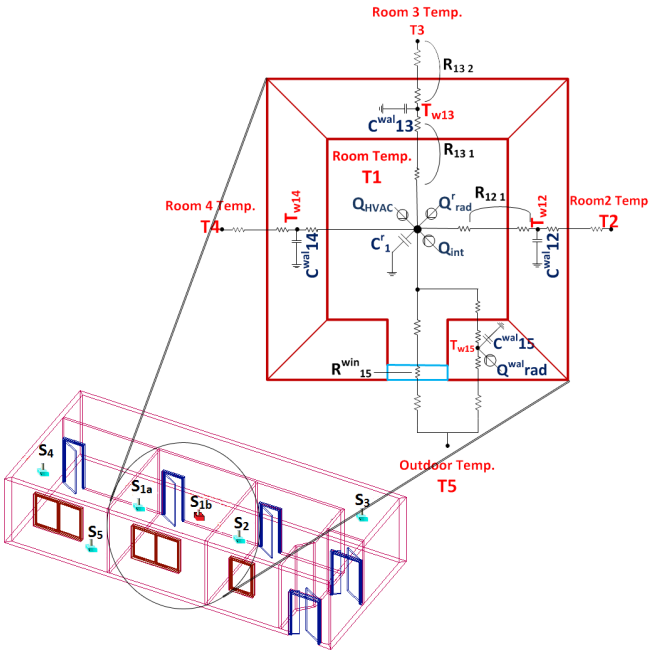

Fig. 1. Resistor-capacitor representation of a typical room with a window.

$Q_{rad_{i,j}}$ is the radiative heat flux density on $w_{i,j}$. $\mathcal{N}_{w_{i,j}}$ is the set of all nodes neighboring $w_{i,j}$. $r_{i,j}$ 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 the set of neighboring *room* nodes for room i. Further details on this thermal model can be found in [21].

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

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

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. air mass flow and supply air temperature). The vector d_t stores the estimated disturbance values, aggregating various unmodeled dynamics such as T_{out}, $\dot{Q}_{int}$ and Q_{rad}, and can be estimated using historical data [21]. $y_t \in \mathbb{R}^m$ is the output vector, representing the temperature of the thermal zones, and C is a constant matrix of proper dimension.

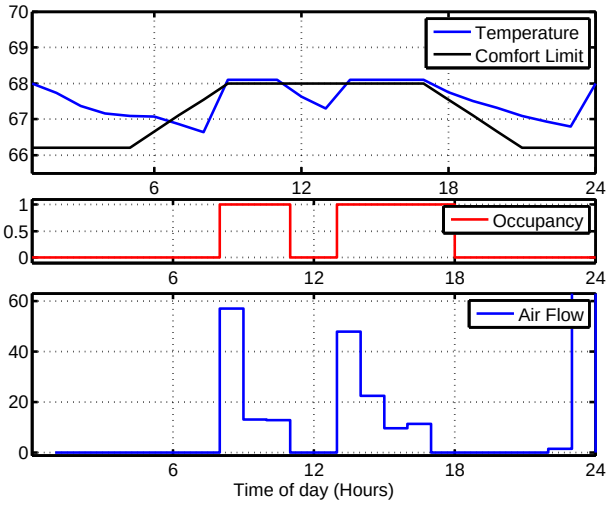

Fig. 2. Room temperature control with constraints based on occupancy, expressed in STL.

C. MPC for Building Climate Control

We consider a commercial building that has an HVAC system controlled by an MPC. We adopt the MPC formulation proposed by Maasoumy et al. [19], with the objective of minimizing the total energy cost (in dollar value). Denote by τ the length of each time slot, and by H the prediction horizon (in number of time slots) of the MPC. Assume that the system dynamics are also discretized with 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:

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

The STL formula was encoded using the robust MILP encoding and results are presented in Figure 2. 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. Discussion

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 building-level 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.

Future work will further explore synthesis in an MPC framework for unbounded STL properties. As mentioned in Section VI-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, which we intend to explore. We have already 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.

References

[1] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[2] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying industrial hybrid systems with MathSAT. *Electronic Notes in Theoretical Computer Science*, 119(2):17 – 32, 2005. Proceedings of the 2nd International Workshop on Bounded Model Checking (BMC 2004) Bounded Model Checking 2004.

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

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

[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. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.

[7] A. Donzé, T. Ferrère, and O. Maler. Efficient robust monitoring for stl. In *CAV*, pages 264–279, 2013.

[8] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In K. Chatterjee and T. A. Henzinger, editors, *FORMATS*, volume 6246 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2010.

[9] 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.

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

[11] M. Franzle and C. Herde. Efficient proof engines for bounded model checking of hybrid systems. *Electronic Notes in Theoretical Computer Science*, 133(0):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.

[12] 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.

[13] 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)*, volume 4763 of *Lecture Notes in Computer Science*, pages 241–256, 2007.

[14] 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.

[15] 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*, pages 2117–2122, Dec 2008.

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

[17] J. Lfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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*, pages 672–677, Dec 2005.

[25] 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.

[26] 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*, CyPhy'14, pages 52–55, New York, NY, USA, 2014. ACM.

[27] 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.

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