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 model predictive control. We present experimental results for controller synthesis on simplified models of a smart micro-grid and HVAC system.

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 [7] and aircraft power system design [19]. 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) [16] 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 robustness of the satisfaction or violation. Such semantics have been defined for STL [6] to assess the robustness of the systems to parameter or timing variations.

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. Seshia are with the Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA 94720, USA donze@berkeley.edu, mehd1@me.berkeley.edu, alberto@berkeley.edu, sseshia@eecs.berkeley.edu

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 [17].

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 [12] 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.

Our work extends the standard Bounded Model Checking (BMC) paradigm for finite discrete systems [3] 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 [18], [9] or by extending SAT solvers to reason about linear inequalities [2], [8]. Similarly, MILP encodings inspired by BMC have been used to generate trajectories for continuous systems with linear temporal logic specifications [10], [11], [20]. However, this is the first work to consider a BMC approach to STL synthesis.

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 for controller synthesis on simplified models of a smart building-level micro-grid and Heating Ventilation and Air Conditioning (HVAC) system. 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 proposed methodology.

II. PRELIMINARIES

A. Discrete-Time Continuous Systems

We consider discrete-time continuous systems of the form

\[ x_{t+1} = f(x_t, u_t) \]  

(1)

where \( t = 0, 1, \ldots \) are the time indices, \( x \in \mathcal{X} \subseteq (\mathbb{R}^n_x \times \{0, 1\}^m_u) \) are the continuous and binary/logical states, \( u \in \mathcal{U} \subseteq (\mathbb{R}^m_u \times \{0, 1\}^m_l) \) are the continuous and logical control inputs, and \( x_0 \in \mathcal{X} \) is the initial state. A run \( \pi = x_0 x_1 x_2 \ldots \) 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 \( t = 0, 1, \ldots \), there exists a control input \( u_t \in \mathcal{U} \) such that \( x_{t+1} = f(x_t, u_t) \). Given an initial state \( x_0 \) and a control input sequence \( u = u_0 u_1 u_2 \ldots \), the resulting run \( \pi = x(0, u) \) of a system modeled by (1) is unique.

Restricting to finite sequences, given a control input sequence \( u_0^N = u_0 u_1 u_2 \ldots u_N \), we let the resulting horizon-\( N \) run be \( \pi = x(x_0, u_0^N) = x_0 x_1 x_2 \ldots x_N \). We also introduce the notion of a generic cost function \( J(x, u) \) that maps finite and infinite runs to \( \mathbb{R} \cup \infty \).

B. Signal Temporal Logic

For this work, 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 size linear in the original size. STL formulas are defined recursively as:

\[ \varphi ::= \mu \ | \ \neg \mu \ | \ \varphi \land \psi \ | \ \varphi \lor \psi \ | \square_{[a,b]} \psi \ | \ \varphi \ U_{[a,b]} \psi \]

where \( \mu \) is a predicate which value is determined by the sign of some function of an underlying signal \( x \), i.e., \( \mu \equiv \mu(x) > 0 \) and \( \psi \) is an STL formula. Additionally, we define \( \Diamond_{[a,b]} \varphi = \top \ U_{[a,b]} \varphi \). The validity of a formula \( \varphi \) with respect to signal \( x \) at time \( t \) is defined inductively as follows:

\[
\begin{align*}
(x, t) &\models \mu \iff \mu(x(t)) > 0 \\
(x, t) &\models \neg \mu \iff \neg \mu((x, t)) \\
(x, t) &\models \varphi \land \psi \iff (x, t) \models \varphi \land (x, t) \models \psi \\
(x, t) &\models \varphi \lor \psi \iff (x, t) \models \varphi \lor (x, t) \models \psi \\
(x, t) &\models \square_{[a,b]} \varphi \iff \forall t' \in [a, t, b], (x, t') \models \varphi \\
(x, t) &\models \varphi \ U_{[a,b]} \psi \iff \exists t' \in [a, t, b] \text{ s.t. } (x, t') \models \psi \\
& \qquad \land \forall t'' \in [t', t''], (x, t'') \models \varphi.
\end{align*}
\]

A run \( \pi = x_0 x_1 x_2 \ldots \) satisfies \( \varphi \), denoted by \( x \models \varphi \), if \( (x, 0) \models \varphi \). Informally, \( x \models \Diamond_{[a,b]} \varphi \) if the property defined by \( \varphi \) holds at every time step between \( a \) and \( b \), \( x \models \square_{[a,b]} \varphi \) if \( \varphi \) holds at some time step between \( a \) and \( b \), and \( x \models \varphi \ U_{[a,b]} \psi \) if \( \varphi \) holds at every time step before \( \psi \) holds, and \( \psi \) holds at some time step between \( a \) and \( b \). Note that since we deal only with discrete-time systems in this work, the STL formulas we consider refer only to intervals over discrete time values. In fact, considering continuous time formulas renders the satisfiability of STL undecidable, so the discrete time restriction is necessary for our approach.

C. Robust Satisfaction of STL formulas

Quantitative or robust semantics define a real-valued function \( \rho^\varphi \) of \( x \) and \( t \) such that \( (x, t) \models \varphi \iff \rho^\varphi(x(t), t) > 0 \). This can be done from the above semantics in a straightforward manner, by propagating values of the functions associated with each predicates using min and max operators corresponding to the different operators of STL. For example, the robust satisfaction of \( \mu_1 \equiv x - 3 > 0 \) at time 0 is simply \( \rho^\mu_1(x, 0) = x(0) - 3 \). The robust satisfaction of \( \mu_1 \land \mu_2 \) is the minimum \( \rho^\mu_1 \) and \( \rho^\mu_2 \). Temporal operators can be treated as conjunction and disjunctions along the time axis, e.g., the robust satisfaction of \( \varphi = \square_{[0,2]} \mu_1 \) 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:

\[
\begin{align*}
\rho^\mu_1(x, t) &= \mu(x(t)) \\
\rho^{-\mu_1}(x, t) &= -\mu(x(t)) \\
\rho^{\min}(x, t) &= \min(\rho^\varphi(x, t), \rho^\psi(x, t)) \\
\rho^{\max}(x, t) &= \max(\rho^\varphi(x, t), \rho^\psi(x, t)) \\
\rho^\varphi(x, t) &= \min_{t' \in [a, t, b]} (\min_{t'' \in [t', t'']} (\rho^\psi(x, t''))(\rho^\mu_1(x, t''))) \\
\rho^\psi U_{[a,b]}(x, t) &= \max_{t' \in [a, t, b]} (\min_{t'' \in [t', t'']} (\rho^\psi(x, t''))(\rho^\mu_1(x, t''))) \setminus \rho^\varphi(x, t).
\end{align*}
\]

III. PROBLEM STATEMENT

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

Problem 1 (Optimal Controller Synthesis from STL):
Given a system of the form (1), an initial state \( x_0 \), an STL formula \( \varphi \) and a cost function \( J \), compute

\[
\begin{align*}
&\arg\min_{u} \quad J(x(0, u)) \\
&\text{s.t. } x(0, u) \models \varphi,
\end{align*}
\]

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

\[
\begin{align*}
&\arg\min_{u} \quad J(x^H(t, x_0(t), u^H(t), u^I(t))) \\
&\text{s.t. } x(0, u) \models \varphi,
\end{align*}
\]

where \( u = u^H(1) u^I(1) u^I(2) \ldots \)

In Sections IV and VI, we present both an open-loop solution to Problem 1, and a solution to Problems 2 using an MPC formulation. In the absence of an objective function \( J \) on runs of the system, we maximize the robustness of the generated runs with respect to \( \varphi \).
IV. OPEN-LOOP CONTROLLER SYNTHESIS

In order to solve Problem 1, we add STL constraints to an MILP formulation of the open-loop control problem. To do so, first, we represent the system trajectory over the MPC prediction horizon as a finite sequence of states satisfying the model dynamics (1). Then we encode the formula \( \varphi \) with a set of Mixed Integer Linear Program (MILP) constraints.

A. Finite Trajectory Parametrization

The notation used in this paper follows that of [4] and [20], but is the first to extend it to support STL specifications. To allow interpretation of the STL specification over infinite sequences of states, an infinite sequence is represented over this finite horizon by a finite trajectory with a loop. Note that this assumption that the trajectory eventually be periodic renders our approach conservative for general systems of the form (1) with an infinite state space.

In order to specify properties of infinite executions of this finite parametrized trajectory, we will enforce a lasso shape on this finite sequence, requiring a loop that makes some eventual portion of the trajectory periodic. We will then encode the STL formula as mixed integer-linear constraints on this finite trajectory. The constraints from the system model are also included as part of the optimization problem, as with standard MPC. This allows Problem 1 to be solved using a mixed-integer linear program (MILP) solver, which, although NP-hard, can be solved in practice for large problems using modern solvers with sophisticated search heuristics.

We now present some definitions that are common to most approaches based on bounded model checking.

Definition 1: A run \( \mathbf{x} \) is a \((N, l)\)-loop if

\[
\mathbf{x} = (x_0 x_1 \ldots x_{l-1})(x_l \ldots x_N)^\omega,
\]

such that \( 0 < l \leq N \) and \( x_{l-1} = x_N \), where \( (\sigma)^\omega \) denotes infinite repetition of sequence \( \sigma \).

Definition 2: Given a run \( \mathbf{x} \) and a bound \( N \in \mathbb{N} \), \( \mathbf{x} \models_N \varphi \) if \( \mathbf{x} \) is a \((N, l)\)-loop for some \( 0 < l \leq N \) and \( \mathbf{x}((0, 0)) \models \varphi \). We propose to solve Problem 1 by replacing \( \varphi(x_0, u) \models \varphi \) with \( \varphi(x_0, u) \models_N \varphi \), where \( N \) is the length of the parametrized trajectory. To do so, we build a set of MILP constraints that is satisfiable if and only if there exists a trajectory of length \( N \) that satisfies \( \varphi \). The satisfiability of these constraints can be checked using a MILP solver, which also yields a solution to the constraints when feasible. This solution gives us the control input \( u \) desired.

There are several components to the encoding of Problem 1 as a set of MILP constraints; these include system constraints, loop constraints and STL constraints.

B. Constraints on system evolution

The system constraints encode valid trajectories of length \( N \) for a system of the form (1) – these constraints hold if and only if the trajectory \( \mathbf{x}(x_0, u) \) satisfies (1) for \( t = 0, 1, \ldots, N \). 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 building-level smart-grid control system model presented in [14]. Other useful examples include mixed-logical dynamical systems such as those presented in [20].

C. Loop constraints for trajectory parametrization

The loop constraints enforce the existence of a loop in the finite system trajectory. We introduce \( N \) binary variables \( l_1, \ldots, l_N \), which determine where the loop forms. These are constraints such that only one is enabled (set to True) at a time, and if \( l_j = 1 \), then \( x_{j-1} = x_N \). The following constraints (which employ the “big M” method from operations research) enforce these requirements:

- \( \sum_{i=1}^{N} l_i = 1 \)
- \( x_N = x_{j-1} + M_j(1 - l_j), j = 1, \ldots, N, \)
- \( x_N \geq x_{j-1} + M_j(1 - l_j), j = 1, \ldots, N, \)

where \( M_j \) are sufficiently large positive numbers.

D. STL constraints

Given a formula \( \varphi \), we introduce a variable \( z_{i,t}^{\varphi} \), whose value is tied to a set of mixed integer linear constraints required for the satisfaction of \( \varphi \) at position \( t \) in the state sequence. In other words, \( z_{i,t}^{\varphi} \) has an associated set of MILP constraints such that \( z_{i,t}^{\varphi} = 1 \) if and only if \( \varphi \) holds at position \( t \). We recursively generate the MILP constraints corresponding to \( z_{i,t}^{\varphi} \) – the value of this variable determines whether a formula \( \varphi \) 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 \( z_{i,t}^{\mu} \in \{0, 1\} \) for times \( t = 0, 1, \ldots, N \). The following constraints enforce that \( z_{i,t}^{\mu} = 1 \) if and only if \( \mu(x_t) > 0 \):

\[
\begin{align*}
\mu(x_t) & \leq M_i z_{i,t}^{\mu} + \epsilon_t \\
-\mu(x_t) & \leq M_i (1 - z_{i,t}^{\mu}) - \epsilon_t
\end{align*}
\]

where \( M_i \) are sufficiently large positive numbers, and \( \epsilon_t \) are sufficiently small positive numbers that serve to bound \( \mu(x_t) \) away from 0. Note that \( z_i = 1 \) 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-D.1, each predicate \( \mu \) has an associated binary variable \( z_{i,t}^{\mu} \) which equals 1 if \( \mu \) holds at time \( t \), and 0 otherwise. In fact, by the recursive definition of our MILP constraints on STL formulas, we can assume that each operand \( \varphi \) in a boolean operation has a corresponding (binary or continuous) variable \( z_{i,t}^{\varphi} \) which is 1 if \( \varphi \) holds at \( t \) and 0 if not. Here we define boolean operations on these variables; these are the building blocks of our recursive encoding.

Given a formula \( \psi \) containing a boolean operation, we add new continuous variables \( z_{i,t}^{\psi} \in [0, 1] \) to represent its truth value at each time step of the parametrized trajectory. These variables are constrained such that \( z_{i,t}^{\psi} = 1 \) if \( \psi \) holds at time \( t \) and \( z_{i,t}^{\psi} = 0 \) otherwise.

Negation: \( \psi = \neg \mu \)

\[ z_{i,t}^{\psi} = 1 - z_{i,t}^{\mu} \]

Conjunction: \( \psi = \land_{i=1}^{m} \varphi_i \)

\[ z_{i,t}^{\psi} \leq z_{i,t}^{\varphi_i}, i = 1, \ldots, m, \]

\[ z_{i,t}^{\psi} \geq 1 - m + \sum_{i=1}^{m} z_{i,t}^{\varphi_i} \]
Disjunction: \( \psi = \lor_{i=1}^{m} \varphi_i \)  
\( z_t^\psi \geq z_t^{\varphi_i}, i = 1, \ldots, m, \)
\( z_t^\psi \leq \sum_{i=1}^{m} z_t^{\varphi_i}. \)

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 \( a_i = \min(t+a, N) \) and \( b_i = \min(t+b, N) \)
Define \( z_t^\psi = \bigwedge_{a_i \leq t \leq b_i} z_t^{\varphi_i} \wedge (\forall j \in \mathbb{N}, j \neq t \wedge \bigwedge_{i=j}^{i=b_i} z_t^{\varphi_i}) \)
The logical operation \( \wedge \) on the variables \( z_t^{\varphi_i} \) here is as defined in Section IV-D.2.
Eventually: \( \psi = \diamond_{[a,b]} \varphi \)
Define \( z_t^\psi = \bigvee_{a_i \leq t \leq b_i} z_t^{\varphi_i} \wedge (\forall j \in \mathbb{N}, j \neq t \wedge \bigvee_{i=j}^{i=b_i} z_t^{\varphi_i}) \)
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 [5]:
\( \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 = \Box_{[0,a]} \varphi_1 \wedge \diamond_{[a,b]} \varphi_2 \wedge \diamond_{[a,b]} (\varphi_1 \mathcal{U} \varphi_2) \)

We will use the linear encoding of the unbounded \( \mathcal{U} \) from [4], including the auxiliary encoding that prevents the pitfalls of circular reasoning on the finite parametrization of the infinite sequences. The interested reader is referred to [4] for the details of why this auxiliary encoding is required.

The auxiliary encoding of the unbounded until is
\( \langle \langle \varphi_1 \mathcal{U} \varphi_2 \rangle \rangle_t = \begin{cases} 
    z_t^{\varphi_1} \wedge \langle \langle \varphi_1 \mathcal{U} \varphi_2 \rangle \rangle_{t+1}, & t = 1, \ldots, N - 1 \\
    z_t^{\varphi_2}. &
\end{cases} \)

With this, we define
\( z_t^{\varphi_1} \mathcal{U} \varphi_2 = z_t^{\varphi_2} \lor (z_t^{\varphi_1} \wedge z_{t+1}^{\varphi_1} \mathcal{U} \varphi_2) \)
for \( t = 1, \ldots, N - 1, \) and
\( z_N^{\varphi_1} \mathcal{U} \varphi_2 = z_N^{\varphi_2} \lor (z_N^{\varphi_1} \wedge (\forall j = 1 \ldots N \wedge \langle \langle \varphi_1 \mathcal{U} \varphi_2 \rangle \rangle_j)). \)

Given this encoding of the unbounded until and the encodings of \( \Box_{[a,b]} \) and \( \diamond_{[a,b]} \) above, we can encode
\( z_t^{\varphi_1} \mathcal{U}_{[a,b]} \varphi_2 = z_t^{[a,b]} \varphi_1 \lor z_t^{[a,b]} \varphi_2 \lor z_t^{[a,b]} (\varphi_1 \mathcal{U} \varphi_2) \).

By induction on the structure of STL formulas \( \varphi, z_t^\varphi = 1 \) if and only if \( \varphi \) holds on the system at time \( t \). With this motivation, given a specification \( \varphi \), we add a final constraint:
\( z_0^\varphi = 0. \)

The union of the STL constraints, system constraints and loop 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. Finally, loop constraints introduce \( N \) additional binary variables.

V. ROBUSTNESS-BASED CODING

The robustness of satisfaction of the STL specification, as defined in II-C, provides a natural objective for the MILP defined in section IV-D.3, either in the absence of, or as a complement to domain-specific objectives on turns 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-D.3. Given a formula \( \varphi \), we introduce a variable \( r_t^\varphi \), and an associated set of MILP constraints such that \( r_t^\varphi > 0 \) if and only if \( \varphi \) holds at position \( t \). We recursively generate the MILP constraints, such that \( r_t^\varphi \) determines whether a formula \( \varphi \) holds in the initial state. Additionally, we enforce that the value of \( r_t^\varphi = \rho^\varphi(x,t) \).

For each predicate \( \mu \in P \), we now introduce variables \( r_t^\mu \) for time indices \( t = 0, 1, \ldots, N \), and set \( r_0^\mu = \mu(x_0) \).
For \( r_t^\varphi \) where \( \psi \) is a boolean formula, we assume that each operand \( \varphi \) has a corresponding variable \( r_t^\varphi = \rho^\varphi(x,t) \). Then the boolean operations are defined as:
Negation: \( \psi = \neg \mu \)  
\( r_t^\psi = -r_t^\mu \)

Conjunction: \( \psi = \land_{i=1}^{m} \varphi_i \)  
\( \sum_{i=1}^{m} b_t^{\varphi_i} = 1 \)  
\( r_t^\psi \leq r_t^{\varphi_i}, i = 1, \ldots, m \)  
\( r_t^\psi - (1-b_t^\varphi)M \leq r_t^\varphi + (1-b_t^\varphi) \)

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

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

The encoding for bounded temporal operators is defined as in Section IV-D.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 \( \varphi \), this construction yields \( r_t^\varphi > 0 \) if and only if \( \varphi \) is satisfied at time \( t \). Therefore, we can replace the constraints over \( z_t^\varphi \)
in Section IV-D.3 by these constraints that compute the value of \( r_0^r \), and instead of (2), add the constraint \( r_0^r > 0 \).

The advantage of this encoding is that it allows us to maximize the value of \( r_0^r \), 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 \( r_0^r > -\epsilon \) for some \( \epsilon > 0 \), thereby easily modifying the problem to allow a limited violation of the STL property.

The disadvantage is that it is significantly more expensive to compute, due to 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, allowing 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 \( \varphi \).

A. Bounded-time STL formulas

The length of the horizon \( H \) is set based on the formula \( \varphi \). To pick \( H \), we compute the sum for each set of nested upper bounds on the temporal operators, and then compute the maximum over this value. This provides a conservative bound on the length of the trajectory required to decide the satisfiability of the formula \( \varphi \). For example, if the STL formula is \( \square [0,10] [1,6] \varphi \), then we require \( H \geq 10 + 6 = 16 \) in order to determine whether the formula is satisfiable. \( H \) can be computed in time linear in the length of the formula.

At time step 0, we will synthesize control \( u^H_0 \) 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 \( u^H_0 \). At the next step of the MPC, we will solve for \( u^H_1 \), while constraining the previous values of \( x_0, u_0 \) in the MILP, 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 \( u^H_t \) 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. \( \varphi = \square (\varphi_{MPC}) \) for bounded STL formulas \( \varphi_{MPC} \). For such formulas, at each step of the MPC computation, we will search for a finite trajectory of horizon length \( H \) (determined from \( \varphi_{MPC} \) as in Section VI-A) that satisfies \( \varphi_{MPC} \).

VII. Case Study I: Building Climate Control

In this case study we consider the problem of controlling building indoor climate, using the model proposed in [15].

A. Building Mathematical Model

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 \( T_{w_{ij}} \) and \( T_{w_{ij}} \), respectively. The temperature of \( w_{ij} \) and \( r_i \) room are governed by the following equation:

\[
\begin{align*}
C_{w_{ij}}^d \frac{dT_{w_{ij}}}{dt} &= \sum_{k \in N_{w_{ij}}} \frac{T_k - T_{w_{ij}}}{R_{w_{ik}}} + r_{w_{ij}} \alpha_{w_{ij}} A_{w_{ij}} Q_{rad_{w_{ij}}} \\
C_i^d \frac{dT_{r_i}}{dt} &= \sum_{k \in N_{r_i}} \frac{T_k - T_{r_i}}{R_{r_{ik}}} + \tilde{m}_{r_i} c_a (T_{s_i} - T_{r_i}) + w_{i} \tau_{w_{i}} A_{w_{i}} Q_{rad_{i}} + Q_{int_{i}},
\end{align*}
\]

where \( C_{w_{ij}}^d \), \( \alpha_{w_{ij}} \) and \( A_{w_{ij}} \) are heat capacity, a radiative heat absorption coefficient, and the area of \( w_{ij} \), respectively. \( R_{w_{ik}} \) is the total thermal resistance between the centerline of wall \( (i, j) \) and the side of the wall on which node \( k \) is located. \( Q_{rad_{w_{ij}}} \) is the radiative heat flux density on \( w_{ij} \). \( N_{w_{ij}} \) is the set of all nodes neighboring \( w_{ij} \), \( r_i \) is a wall identifier, which equals \( 0 \) for internal walls and \( 1 \) for peripheral walls (where either \( i \) or \( j \) is the outside node). \( T_{r_i}, C_i^d \) and \( \tilde{m}_{r_i} \) 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, \( \tau_{w_{i}} \) is the transmissivity of the glass of window \( i \), \( A_{w_{i}} \) 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 \( Q_{int_{i}} \) is the internal heat generation in room \( i \). \( N_{r_{i}} \) is the set of all of the neighboring room nodes to room \( i \). More details on this thermal model can be found in [15].

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

\[
\dot{x}_t = f(x_t, u_t, d_t), \quad y_t = C x_t
\]

Here \( x_t \in \mathbb{R}^n \) is the state vector representing the temperature of the nodes in the thermal network, and \( u_t \in \mathbb{R}^m \) 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 disturbance values, aggregating various unmodeled dynamics such as \( T_{out}, Q_{int} \) and \( Q_{rad} \), and can be estimated using historical data [15]. \( y_t \in \mathbb{R}^m \) is the output vector, representing the temperature of the thermal zones, and \( C \) is a constant matrix of proper dimension.
of the power system presented in [14]. The interconnection encoding and results are presented in Figure 2.

The purpose of the MPC is to maintain a comfort temperature given by $T_{\text{conf}}$ whenever the room is occupied while minimizing the cost of heating. These constraints can be expressed with the following problem:

$$\min_{u_t} \sum_{k=0}^{H-1} \|u_{t+k}\|$$  \text{s.t.}

$$x_{t+k+1} = f(x_{t+k}, u_{t+k}, d_{t+k}),$$

$$x_t = \varphi \text{ with } \varphi = \square_{[0,H]}(\text{occ}_t > 0) \Rightarrow (T_t > T_{\text{conf}})$$

$$u_{t+k} \in U_{t+k}, \quad k = 0, \ldots, H-1$$

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

B. 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 in [13], with the objective of minimizing the total energy cost (in dollar value). Denote by $\tau$ 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 $\tau$. Here we consider $\tau = 1$ hr and $H = 24$ hrs.

At each time $t$, the predictive controller solves an optimal control problem to compute $\bar{u}_t = [u_1, \ldots, u_{t+H-1}]$, and minimizes the cumulative norm of $u_t$: $\sum_{k=0}^{H-1} \|u_{t+k}\|$. We assume known an occupancy function $\text{occ}_t$ 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 $T_{\text{conf}}$ whenever the room is occupied while minimizing the cost of heating. These constraints can be expressed with the following problem:

$$\min_{u_t} \sum_{k=0}^{H-1} \|u_{t+k}\|$$  \text{s.t.}

$$x_{t+k+1} = f(x_{t+k}, u_{t+k}, d_{t+k}),$$

$$x_t = \varphi \text{ with } \varphi = \square_{[0,H]}(\text{occ}_t > 0) \Rightarrow (T_t > T_{\text{conf}})$$

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

VIII. CASE STUDY II: REGULATION CONTROL FOR SMART GRID

The second case study we consider is the smart grid model of the power system presented in [14]. The interconnection of power system components, including a governor, turbine and generator, is shown in the block diagram in Figure 3. In the diagram, $\delta P_C$ is a control input which acts against an increase or decrease in power demand to regulate the system frequency, and $\delta P_D$ denotes fluctuations in power demand, modeled as an exogenous input (disturbance).

A. Two-Area System Model

We consider a two-area interconnected system with two buses connected by a tie line with reactance $X_{\text{tie}}$. Power flow from area 1 to area 2 is denoted by $P_{\text{tie}}$. A positive $\delta P_{\text{tie}}$ represents an increase in power transfer from area 1 to area 2. This is equivalent in effect to increasing the load of area 1 and decreasing the load of area 2. Each area consists of the subsystems shown in Figure 3. Next, we present the mathematical model of the two-area system. Note that for states, $x$, the superscript refers to the control area (i.e., $i = 1, 2$), and the subscript indexes the state in each area. In this example we show time in brackets $[]$.

$$\begin{align}
\frac{dx_1^1}{dt} &= (-D^1 x_1^1 + \delta P_{\text{tie}} - \delta P_{\text{tie}} - \delta P_{\text{tie}} + \delta P_{\text{tie}}) \\
\frac{dx_2^1}{dt} &= (x_2^1 - x_2^2) \\
\frac{dx_2^2}{dt} &= (P_{\text{tie}} x_1^1 - x_2^2) \\
\frac{dx_3^1}{dt} &= (x_2^1 - x_2^1) \\
\frac{dx_3^2}{dt} &= (x_2^1 - x_2^1) \\
\frac{dx_4^1}{dt} &= (x_2^1 - x_2^1) \\
\frac{dx_4^2}{dt} &= (x_2^1 - x_2^1)
\end{align}$$

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

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

Fig. 3. 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 [14].
where $\delta P^m_i$ and $P^i_{GV}$ are given by $\delta P^m_i = K^1_i x^1_i + K^2_i x^2_i$, and $P^i_{GV} = (1 - T_2/T_3)x^0_i + (T_2/T_3)x^1_i$. $D$ 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 points of the turbine. In formulation (8), the first state represents the frequency increment, $x^1_i = \delta \omega_i$. All state dynamics are derived using the mathematical model of each subsystem, as presented in [14]. The state space model (8) can be discretized and written in compact form as

$$x[k+1] = Ax[k] + B_A u_{sec}[k] + B_2 u_{anc}[k] + Ed[k]. \quad (9)$$

We use this state update equation in Section VIII-C, where we present the MPC formulation. Input signals are $u_{sec} = [\delta P^1_C \delta P^2_C]^T$, the ancillary inputs are $u_{anc} = [\delta P^1_{anc} \delta P^2_{anc}]^T$, and the exogenous inputs (i.e. disturbances or variations in demands) are denoted by $d = [\delta P^1_D \delta P^2_D]^T$.

B. Automatic Generation Control

In the classical AGC, a simple PI control is utilized to regulate the grid frequency. The Area Control Error (ACE) is defined as $ACE^i = \delta P^i_{tie} + \beta^i x^1_i$, where $\delta P^i_{tie} = P^i_{tie} - P^i_{tie,scheduled}$, and $\beta^i$ is the bias coefficient of area $i$. The standard industry practice is to set the bias $\beta^i$ 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 ($\delta P^i_C$). In other words, the control input $\delta P^i_C$ is given by $\delta P^i_C = -K^i x^1_i$, where $K^i$ is the feedback gain and $\frac{dx^1_i}{dt} = ACE^i$. We propose controller synthesis for the ancillary services, complementing the primary control of AGC, as described in VIII-C.

C. MPC for Ancillary Services

We present an MPC scheme to control the ancillary service to improve on the classical AGC practice. This optimization-based control framework is utilized as a higher-level control in a “hierarchical” scheme on top of the low-level classical AGC control [14]. We require that $u_{anc}$ satisfies $u_{anc} \leq u_{anc}[k+j] \leq \bar{u}_{anc}$ for some $\underline{u}_{anc} < 0$ and $\bar{u}_{anc} > 0$, and a maximum ramp constraint:

$$|u_{anc}[k+1] - u_{anc}[k]| \leq \lambda, \text{for some } \lambda > 0. \quad (10)$$

The notation $u_{anc}[k+j]$ denotes that predictions of $u_{anc}$ for future times $k+j$ are obtained at each time step $k$. For ease of notation, we drop the $k$ from here on. At each time step $k$, we thus solve the following problem:

$$\min_{u_{anc}[k]} J(ACE, U_{anc}) + ||x[k + H] - x_{ref}||_Q \quad \text{s.t.}$$

$$x[k+j+1] =$$

$$Ax[k+j] + B_2 u_{anc}[k+j] + Ed[k+j]$$

$$u_{anc} \leq u_{anc}[k+j] \leq \bar{u}_{anc}$$

$$|u_{anc}[k+j+1] - u_{anc}[k+j]| \leq \lambda$$

$$x[k+H] \in \mathcal{X}[H]$$

where $U_{anc}[k] = [u_{anc}[k], u_{anc}[k+1], \ldots, u_{anc}[k+H-1]]$ is the vector of inputs from $k$ to $k+H$ and $H$ is the prediction horizon. All the constraints of problem (11) that depend on $j$ should hold for $j = 0, 1, \ldots, H - 1$. $x_{ref}$ is a reference state and $\mathcal{X}[H]$ is the terminal constraint on states. The second term in the cost function and the last constraint are terminal cost and terminal constraints, introduced to guarantee feasibility and stability of the receding horizon control implementation.

The cost function proposed in [14] minimizes the $\ell_2$ norm of the ACE signal in areas $i = 1, 2$, 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 $J(ACE, U_{anc}) = ||U_{anc}||_{\ell_2} = \sum_{i=1}^{H-1} \sum_{j=0}^{H-1} (U_{anc}(k+j))^2$, and an STL formula $\varphi$ which says that whenever $|ACE^1|$ is larger than 0.01, it should become less than 0.01 in less than $\tau$ s. More precisely we used $\varphi = \Box(\varphi_t)$ with

$$\varphi_t = (\neg(|ACE^1| < .01)) \land (\neg(|ACE^2| < .01))$$

where $U_{anc}[k] = [u_{anc}[k], u_{anc}[k+1], \ldots, u_{anc}[k+H-1]]$ is the vector of inputs from $k$ to $k+H$ and $H$ is the prediction horizon. All the constraints of problem (11) that depend on $j$ should hold for $j = 0, 1, \ldots, H - 1$. $x_{ref}$ is a reference state and $\mathcal{X}[H]$ is the terminal constraint on states. The second term in the cost function and the last constraint are terminal cost and terminal constraints, introduced to guarantee feasibility and stability of the receding horizon control implementation.

The cost function proposed in [14] minimizes the $\ell_2$ norm of the ACE signal in areas $i = 1, 2$, 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 $J(ACE, U_{anc}) = ||U_{anc}||_{\ell_2} = \sum_{i=1}^{H-1} \sum_{j=0}^{H-1} (U_{anc}(k+j))^2$, and an STL formula $\varphi$ which says that whenever $|ACE^1|$ is larger than 0.01, it should become less than 0.01 in less than $\tau$ s. More precisely we used $\varphi = \Box(\varphi_t)$ with

$$\varphi_t = (\neg(|ACE^1| < .01)) \land (\neg(|ACE^2| < .01))$$

where $U_{anc}[k] = [u_{anc}[k], u_{anc}[k+1], \ldots, u_{anc}[k+H-1]]$ is the vector of inputs from $k$ to $k+H$ and $H$ is the prediction horizon. All the constraints of problem (11) that depend on $j$ should hold for $j = 0, 1, \ldots, H - 1$. $x_{ref}$ is a reference state and $\mathcal{X}[H]$ is the terminal constraint on states. The second term in the cost function and the last constraint are terminal cost and terminal constraints, introduced to guarantee feasibility and stability of the receding horizon control implementation.

The cost function proposed in [14] minimizes the $\ell_2$ norm of the ACE signal in areas $i = 1, 2$, 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 $J(ACE, U_{anc}) = ||U_{anc}||_{\ell_2} = \sum_{i=1}^{H-1} \sum_{j=0}^{H-1} (U_{anc}(k+j))^2$, and an STL formula $\varphi$ which says that whenever $|ACE^1|$ is larger than 0.01, it should become less than 0.01 in less than $\tau$ s. More precisely we used $\varphi = \Box(\varphi_t)$ with

$$\varphi_t = (\neg(|ACE^1| < .01)) \land (\neg(|ACE^2| < .01))$$

where $U_{anc}[k] = [u_{anc}[k], u_{anc}[k+1], \ldots, u_{anc}[k+H-1]]$ is the vector of inputs from $k$ to $k+H$ and $H$ is the prediction horizon. All the constraints of problem (11) that depend on $j$ should hold for $j = 0, 1, \ldots, H - 1$. $x_{ref}$ is a reference state and $\mathcal{X}[H]$ is the terminal constraint on states. The second term in the cost function and the last constraint are terminal cost and terminal constraints, introduced to guarantee feasibility and stability of the receding horizon control implementation.

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