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] 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 synthesized over a Buchi automaton for the specification. We circumvent these expensive computations of a finite state abstraction of the system and a Buchi automaton 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 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 micro-grid 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

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

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, \( u \in U \subseteq (\mathbb{R}^{n_u} \times \{0, 1\}^{m_u}) \) are the (continuous and logical) control inputs, and \( x_0 \in \mathcal{X} \) is the initial state. A run \( x = x_0 x_1 x_2 \ldots \) 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 \( u_t \in U \) such that \( x_{t+1} = f(x_t, u_t) \). Given an initial state \( x_0 \) and a control input sequence \( u^N = u_0 u_1 u_2 \ldots u_{N-1} \), the resulting horizon- \( N \) run of a system modeled by (1), which we denote by \( x(x_0, u^N) = x_0 x_1 x_2 \ldots x_N \) is unique. Note that a horizon- \( N \) run as defined here has \( N + 1 \) states. We also introduce a generic cost function \( J(x, u) \) 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:

\[ \varphi ::= \mu | \neg \mu | \varphi \land \psi | \varphi \lor \psi | \Box_{[a, b]} \psi | \varphi \mathcal{U}_{[a, b]} \psi \]

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

\[
\begin{align*}
(x, t) &\models \mu \quad \iff \mu(x(t)) > 0 \\
(x, t) &\models \neg \mu \quad \iff \neg((x, t) \models \mu) \\
(x, t) &\models \varphi \land \psi \quad \iff (x, t) \models \varphi \land (x, t) \models \psi \\
(x, t) &\models \varphi \lor \psi \quad \iff (x, t) \models \varphi \lor (x, t) \models \psi \\
(x, t) &\models \Box_{[a, b]} \psi \quad \iff \forall t' \in [t + a, t + b], (x, t') \models \psi \\
(x, t) &\models \varphi \mathcal{U}_{[a, b]} \psi \quad \iff \exists t' \in [t + a, t + b] \text{ s.t. } (x, t') \models \psi \land \forall t'' \in [t, t'), (x, t'') \models \varphi.
\end{align*}
\]

A signal \( x = x_0 x_1 x_2 \ldots \) satisfies \( \varphi \), denoted by \( x \models \varphi \), if \( (x, 0) \models \varphi \). Informally, \( x \models \Box_{[a, b]} \psi \) if the property defined by \( \varphi \) holds at every time step between \( a \) and \( b \), and \( x \models \varphi \mathcal{U}_{[a, b]} \psi \) if \( \varphi \) holds at every time step before \( \psi \) holds, and \( \psi \) holds at some time step between \( a \) and \( b \). Additionally, we define \( \Box_{[0, 10]} \varphi = \top \mathcal{U}_{[0, 10]} \varphi \), so that \( x \models \Box_{[0, 10]} \varphi \) if \( \varphi \) 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 \( \varphi \) is bounded-time if it contains no unbounded operators. The bound of a bounded formula \( \varphi \) 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 \( \varphi \). 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 \( \rho^\varphi \) of signal \( x \) and \( t \) such that \( (x, t) \models \varphi \equiv \rho^\varphi(x, t) > 0 \). 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 \( \mu_1 \land \mu_2 \) is the minimum of \( \rho^\mu_1 \) and \( \rho^\mu_2 \). 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:

\[
\begin{align*}
\rho_1(x, t) &= \mu(x(t)) \\
\rho_{-\mu}(x, t) &= -\mu(x(t)) \\
\rho_{\varphi \land \psi}(x, t) &= \min(\rho_\varphi(x, t), \rho_\psi(x, t)) \\
\rho_{\varphi \lor \psi}(x, t) &= \max(\rho_\varphi(x, t), \rho_\psi(x, t)) \\
\rho_{\varphi \lor \psi}(x, t) &= \min_{t \in [a, a+b]} \rho_\varphi(x, t') \\
\rho_{\varphi \land \psi}(x, t) &= \min_{t \in [a, a+b]} \rho_\psi(x, t') \\
\end{align*}
\]

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 \(\varphi\) and cost function \(J\), compute

\[
\arg\min_{u_N} J(x(0, u^N)) \quad \text{s.t.} \quad x(0, u^N) = \varphi
\]

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

\[
\arg\min_{u_{H,t}} J(x(t, u^H.t), u^{H,t}) \quad \text{s.t.} \quad x(0, u) = \varphi,
\]

where \(H\) is a finite horizon provided as a user input or selected in some other fashion, \(u^{H,t}\) is the horizon-\(H\) control input computed at each time step \(t\), and \(u = u^0_0, u^0_1, u^0_2, \ldots\).

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 \(\varphi\). 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 \(x(0, u_N)\) 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. 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 \(\varphi\), we introduce a variable \(z^\varphi_t\), 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 of horizon \(N\). In other words, \(z^\varphi_t\) has an associated set of MILP constraints such that \(z^\varphi_t = 1\) if and only if \(\varphi\) holds at position \(t\). We recursively generate the MILP constraints corresponding to \(z^\varphi\) – 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 \(z_\mu^t \in \{0, 1\}\) for times \(t = 0, 1, \ldots, N\). The following constraints enforce that \(z_\mu^t = 1\) if and only if \(\mu(x_t) > 0\):

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

where \(M_\mu\) is sufficiently large positive numbers, and \(\epsilon_t\) are sufficiently small positive numbers that serve to bound \(\mu(x_t)\) away from 0. Note that \(z_\mu = 1\) 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 \(\mu\) has an associated binary variable \(z_\mu^t\) 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, each operand \(\varphi\) in a Boolean operation has a corresponding variable \(z_\varphi^t\) 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. The definitions in this subsection are identical to those in [27].

Logical operations on variables \(z^\varphi_t \in [0, 1]\) are defined as follows:

Negation: \(z^\varphi_t = -z_t^\varphi\)

Conjunction: \(z^\varphi_t = \land_{i=1}^m z^\varphi_{ti}\)

Disjunction: \(z^\varphi_t = \lor_{i=1}^m z^\varphi_{ti}\)

Given a formula \(\psi\) containing a Boolean operation, we add new continuous variables \(z^\psi_t \in [0, 1]\), set \(z^\varphi_t = -z^\psi_t\), \(z^\varphi_t = \land_{i=1}^m z^\psi_{ti}\), and \(z^\varphi_t = \lor_{i=1}^m z^\psi_{ti}\) for \(\psi = \land \varphi_i\) and \(\psi = \lor \varphi_i\), respectively. These constraints enforce that \(z^\varphi_t = 1\) if \(\psi\) holds at time \(t\) and \(z^\varphi_t = 0\) otherwise.

3) Temporal constraints: We first present encodings for the \(\square\) and \(\diamond\) operators. We will use these encodings to define the encoding for the \(U_{[a, b]}\) operator.

Always: \(\psi = \square_{[a, b]} \varphi\)

Let \(a^N_t = \min(t + a, N)\) and \(b^N_t = \min(t + b, N)\)

Define \(z^\psi_t = \land_{i=a^N_t}^{b^N_t} z^\varphi_t\)

The logical operation \(\land\) on the variables \(z^\varphi\) here is as defined in Section IV-B.2. Intuitively, this encoding enforces that the formula \(\varphi\) is satisfied at every time step on the interval \([a, b]\) relative to time step \(t\).
Eventually: $\psi = \bigotimes_{i \in [a, b]} \varphi$

Define $z_t^i = \bigotimes_{i=a}^{b} z_t^i$. This encoding enforces that the formula $\varphi$ is satisfied at some time step on the interval $[a, b]$ relative to time step $t$.

Until: $\psi = \varphi_1 \cup \varphi_2$

The bounded until operator $\cup_{[a, b]}$ can be defined in terms of the unbounded $\cup$ (inherited from LTL) as follows [7]:

$\varphi_1 \cup_{[a, b]} \varphi_2 = \bigotimes_{i=a}^{b} \varphi_1 \land \bigotimes_{i=a}^{b} \varphi_2 \land \bigotimes_{i=a}^{b} (\varphi_1 \cup \varphi_2)$

We define

$z_t^{\varphi_1} \cup \varphi_2 = z_t^{\varphi_2} \lor (z_t^{\varphi_1} \land z_{t+1}^{\varphi_1} \cup \varphi_2)$

for $t = 1, \ldots, N - 1$, and

$z_N^{\varphi_1} \cup \varphi_2 = z_N^{\varphi_2}$.

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

$z_t^{\varphi_1} \cup_{[a, b]} \varphi_2 = z_t^{\varphi_1} \land z_t^{\varphi_1} \cup_{[a, b]} \varphi_2 \land z_t^{\varphi_1} \cup_{[a, b]} (\varphi_1 \cup \varphi_2)$.

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

$z_0^i = 1$. (2)

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 is 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 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 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-B.3. Given a formula $\varphi$, we introduce a variable $r_t^i$, and an associated set of MILP constraints such that $r_t^i > 0$ if and only if $\varphi$ holds at position $t$. We recursively generate the MILP constraints, such that $r_0^i$ determines whether a formula $\varphi$ holds in the initial state. Additionally, we enforce that the value of $r_t^i = \rho_\varphi(x, t)$.

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: $r_t^\psi = -r_t^\psi$, $r_t^\psi = -r_t^\psi$

Conjunction: $r_t^\psi = \land_{i=1}^m r_t^{\psi_i}$.

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

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 $\varphi$, this construction yields $r_t^\varphi > 0$ if and only if $\varphi$ is satisfied at time $t$. Therefore, we can replace the constraints over $z_t^i$ in Section IV-B.3 by these constraints that compute the value of $r_t^\varphi$, and instead of (2), add the constraint $r_t^\varphi > 0$.

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 $r_0^\varphi$, 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 $\rho_\varphi^0 > -\epsilon$ 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 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 minima, 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 $\varphi$.

A. Synthesis for bounded-time STL formulas

The length of the horizon $H$ is chosen to be at least the bound of formula $\varphi$. 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 $u^{H,0}_0$. 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 may 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. $\varphi = \mathcal{G}(\varphi_{MPC})$ for bounded STL formulas $\varphi_{MPC}$. For such formulas, at each step of the MPC computation, we will search for a finite trajectory of length $H$ (determined from $\varphi_{MPC}$ as in Section VI-A) that satisfies $\varphi_{MPC}$.

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:

- $\varphi_1 = \mathcal{G}[0,0,1]x_i^{(1)} > 0.1$
- $\varphi_2 = \mathcal{G}[0,0,1](x_i^{(1)} > 0.1) \land \mathcal{G}[0,0,1](x_i^{(2)} < -0.5)$
- $\varphi_3 = \mathcal{G}[0,0,5] \mathcal{G}[0,0,1](x_i^{(3)} > 0.1)$
- $\varphi_4 = \mathcal{G}[0,0,2](x_i^{(1)} > 0.1) \land \mathcal{G}[0,0,1](x_i^{(2)} > 0.1) \land \mathcal{G}[0,0,1](x_i^{(3)} > 0.1))$

In this study, we used the trivial system $x = u$, where $x$ is a 3-dimensional signal (i.e. $x = x_i^{(1)}, x_i^{(2)}, x_i^{(3)}$), so that no constraint is generated for the systems dynamics, and the cost function $J(x,u) = \sum_{k=1}^{N} \|u_k\|_1$. Note that the output of this procedure for a formula $\varphi$ is a signal of minimal norm which satisfies $\varphi$ when using the Boolean encoding and which satisfies $\varphi$ with a specified robustness $\rho^\varphi(x) = 0.1$ 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.\footnote{http://www.gurobi.com/}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline
Formula & #constraints & Yalmip Time (s) & Solver time (s) \\
\hline
$\varphi_1$ & 154,488 & 1.71 & 2.04 & 0.0070 & 0.0085 \\
$\varphi_2$ & 364,897 & 1.94 & 2.69 & 0.0150 & 0.0229 \\
$\varphi_3$ & 244,1282 & 1.84 & 3.15 & 0.0064 & 0.1356 \\
$\varphi_4$ & 574,1330 & 2.29 & 3.37 & 0.2167 & 238.6 \\
\hline
\end{tabular}
\caption{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.}
\end{table}

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 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 $\varphi_4$, 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:

$$\frac{dw_{i,j}}{dt} = \sum_{k \in N_{w_{i,j}}} \frac{T_{r_k} - T_{w_{i,j}}}{R_{i,j,k}} + r_{i,j} \alpha_{i,j} A_{w_{i,j}} Q_{rad_{i,j}}$$

$$\frac{dT_{r_i}}{dt} = \sum_{k \in N_{r_i}} \frac{T_{r_k} - T_{r_i}}{R_{i,k}} + \frac{\dot{m}_{r,i} c_a (T_{w_i} - T_{r_i}) + w_{i,r} \tau_{w_i} A_{w_i} Q_{rad_i} + \dot{Q}_{int_i}}{m_{r_i} c_i}$$

where $C_{i,j}$, $\alpha_{i,j}$ and $A_{w_{i,j}}$ are heat capacity, a radiative heat absorption coefficient, and the area of $w_{i,j}$, respectively.
be estimated using historical data [21].

$R_{i,j,k}$ is the total thermal resistance between the centerline of wall $(i,j)$ and the side of the wall on which node $k$ is located. $Q_{rad,j}$ is the radiative heat flux density on $w_{i,j}$. $N_{w_{i,j}}$ 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). $T_{i}$, $C_{i}$ and $m_{i}$ are the temperature, heat capacity and air mass flow into room $i$. $N_{w}$ is a wall 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 $i$, $A_{win}$ 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 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:

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

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 estimated disturbance values, aggregating various unmodeled dynamics such as $T_{out}$, $Q_{int}$ 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.

$\tau_{w,i}$ is the transmissivity of the glass of window $i$, $A_{win}$ 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 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:

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

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 estimated disturbance values, aggregating various unmodeled dynamics such as $T_{out}$, $Q_{int}$ 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.

$\tau_{w,i}$ is the transmissivity of the glass of window $i$, $A_{win}$ 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 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:

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

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 estimated disturbance values, aggregating various unmodeled dynamics such as $T_{out}$, $Q_{int}$ 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.

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 $\tau$ 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 $\tau = 0.5$ hr and $H = 24$. At each time $t$, the predictive controller solves an optimal control problem to compute $\bar{u}_t = [u_t, \ldots, u_{t+H-1}]$, and minimizes the cumulative norm of $u_t$: $\min_{u_t} \sum_{k=0}^{H-1} \|u_{t+k}\|$. 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 $T_{conf}$ whenever the room is occupied while minimizing the cost of heating. This problem can be expressed as follows:

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

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

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

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

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