Robust Control for Signal Temporal Logic Specifications using Average Space Robustness

Lars Lindemann and Dimos V. Dimarogonas

Abstract—Control systems that fulfill formal specifications are an active research area, especially for motion and task planning applications. Recent methods suffer from state explosion problems which make them inconvenient to use in practice. We propose a framework that can be seen as an alternative approach by avoiding automata representation. The unifying framework consists of Signal Temporal Logic based specifications and a Model Predictive Controller to robustly control linear systems. Novel quantitative semantics for Signal Temporal Logic, called Average Space Robustness, are introduced and directly incorporated into the cost function of the Model Predictive Controller. Consequently, this methodology not only satisfies temporal logic formulas, but also satisfies them as robust as possible. The convex optimization problem encapsulated in this framework can be solved as a linear programming problem. Robustness against disturbances and model uncertainties combined with low computation times are depicted in simulations.

I. INTRODUCTION

Temporal logic control synthesis has extensively been studied during the last years. Current approaches translate a temporal formula into an automata representation, where all accepting runs correspond to the satisfaction of the temporal formula. At the same time, the workspace is abstracted into a transition system. This paper will present an approach where we will not end up with automata representations. By doing so, state explosion problems that these methods suffer from may be avoided.

Branching-time logics and linear-time logics have been under consideration in the literature. Their difference lies within their expressiveness [1]. However, linear-time logics have established their application in control synthesis with a focus on Linear Temporal Logic (LTL) as in [2], [3], [4], [5], [6] and recently also on Metric Interval Temporal Logic (MITL) as in [7]. LTL and MITL formulas can be translated into Büchi automata as explained in [8], [9] and [10], which allows calculating a product automaton with the system under consideration. By using a Depth-First search or a similar search algorithm on this product automaton, a suitable run satisfying the specification can be found. In contrast to MITL, Metric Temporal Logic (MTL) is not decidable as noted in [11] and [11]. Metric time logics extend LTL by adding a real-time notion to make use of quantitative temporal requirements.

The next extension is to equip temporal logics with real-valued signals, i.e. adding a continuous space, instead of using boolean propositions. A logic making use of a continuous space is called Signal Temporal Logic (STL), which was firstly introduced in [12] within the context of monitoring temporal properties of signals. As a consequence, it is possible to measure the satisfaction of a formula in a quantifiable way besides obtaining a boolean answer. Incorporating time control into control theory may be suitable to a variety of applications where temporal patterns are of high importance as for instance in motion and task planning of multi-agent systems. STL is much related to robustness by the introduction of space robustness as in [13]. Without considering STL, [14] already introduced a similar notion of robustness for continuous signals as the distance to the set not fulfilling the specification. STL can not be translated into an automata representation. However, STL can be used for control synthesis together with Model Predictive Control (MPC) as for instance in [15], [16] and [17]. These papers incorporate space robustness into mixed-integer linear programming constraints. In our approach, we put a modified version of space robustness directly into the cost function of an adapted MPC framework.

Model Predictive Control has been under consideration in research and industry for the last 30 years [18]. A MPC incorporates process constraints and the system model into an optimal control problem that uses a finite prediction and control horizon. At each sampling step this optimal control problem is solved, outputting an input sequence. Only the first element of this sequence is applied to the system until the procedure is repeated at the next sampling step. There are many advantages due to the natural inclusion of time and constraints and the easy implementation itself. One drawback is that stability analysis is complicated and in the need of sophisticated methods as described in [19], [20] and [21].

Our contribution can be summarized as follows: First, we introduce a novel space robustness measure, namely Average Space Robustness (ASR). ASR has the advantage that it considers average satisfaction. This is new since space robustness as in [13] rather focus on the worst case scenario, i.e. on the minimum values of a signal within a time interval. While ensuring satisfaction, ASR gives better average satisfaction than space robustness. An analogy can be made by looking at $H_2$ and $H_{\infty}$ control as in [22]. Second, ASR is directly incorporated into the cost function of an MPC framework. This is novel since current approaches as in [15] or [16] only incorporate space robustness into the constraints of a Mixed Integer Linear Program. Hence, here we directly maximize robustness against model uncertainties and disturbances, while...
still assuring satisfaction of the formula since ASR is also placed into the constraints of our MPC. Another advantage of the proposed methodology is that the encapsulated cost function is linear, hence convex, and solvable by using linear programming.

The remainder of the paper is organized as follows: Section II introduces notation, Signal Temporal Logic and the problem statement. Section III presents the problem solution, which is verified in section IV by simulating a double tank process. Results are discussed shortly. Conclusion and outlook are provided in section V.

II. PRELIMINARIES

Throughout the paper scalar quantities are denoted as lowercase, non-bold letters \( x \). Column vectors, containing scalar quantities, are lowercased, bold letters \( \mathbf{x} \) and matrices are denoted as uppercase, non-bold letters \( X \). Time dependence of all quantities is highlighted by \( x(t), \mathbf{x}(t) \) and \( X(t) \). \( \mathbb{R}^n \) is the \( n \)-dimensional vector space over the real numbers \( \mathbb{R} \), whereas \( \mathbb{B} \) and \( \mathbb{N} \) are the sets of boolean values and natural numbers, respectively. \( \textbf{I} \) is a vector only containing ones and of appropriate size. True and false are denoted by \( \top \) and \( \bot \) and \( \otimes \) denotes the Kronecker product. Furthermore, \( \| \mathbf{x} \|_\infty = \sup_t \| x(t) \| \) is the \( L_\infty \)-norm and \( \| \mathbf{x} \|_2 = \sqrt{\int_{-\infty}^{\infty} \| x(t') \|^2 dt'} \) is the \( L_2 \)-norm.

A. Signals and Systems

Let \( x(t), y(t) \) and \( u(t) \) be signals and state, output and input of a system, respectively. We consider linear, time-invariant systems with perfect state measurement of the form

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= x(t).
\end{align*}
\]

The continuous system will be discretized to

\[
\begin{align*}
x(k+1) &= A_dx(k) + B_du(k) \\
y(k) &= x(k),
\end{align*}
\]

where \( A_d \in \mathbb{R}^{n \times n} \) and \( B_d \in \mathbb{R}^{n \times m} \). Later on, we will add an additive noise term \( v(t) \) to this system.

B. Signal Temporal Logic

Signal Temporal Logic is a predicate logic based on signals that represents both continuous time and space. STL was originally introduced in [12] and later on refined in [13]. The predicates used in STL are denoted with \( \mu \) and their truth value is determined by

\[
\mu = \begin{cases} 
\top \text{ if } f(x) > 0 \\
\bot \text{ if } f(x) \leq 0,
\end{cases}
\]

where the sign of a predicate function \( f(x) \) determines the truth value of the predicate. Hence, \( f \) maps from \( \mathbb{R}^n \) to \( \mathbb{R} \) and \( \mu \) maps from \( \mathbb{R}^n \) to \( \mathbb{B} \); \( \mu \) can be an element of the set \( P = \{ \mu_1, \mu_2, \cdots, \mu_{G_n} \} \), where \( G_n \) indicates the number of predicates. STL is inductively defined as in Definition 1 where the temporal operators \( U_{[a,b]} \) (until), \( F_{[a,b]} \) (eventually) and \( G_{[a,b]} \) (always) are already used.

**Definition 1.** The STL syntax is defined as [13]:

1) \( \top \) and \( \bot \) are STL formulas
2) Any propositional variable \( \mu_i \), from \( P \), where \( i \in \mathbb{N} \), is a STL formula
3) Given a STL formula \( \phi \) and \( \psi \), \( \neg \phi \), \( \phi \land \psi \), \( \phi \lor \psi \), \( \phi U_{[a,b]} \psi \), \( F_{[a,b]} \phi \), \( G_{[a,b]} \phi \), \( \phi \equiv \psi \) and \( \phi \iff \psi \) are also STL formulas

The eventually-operator can also be derived as \( F_{[a,b]} \phi = \top U_{[a,b]} \phi \) and the always-operator as \( G_{[a,b]} \phi = \neg F_{[a,b]} \neg \phi \).

The semantics of STL are introduced in Definition 2; [13], [16]:

\[
(x,t) \models \phi \iff f(x(t)) > 0
\]

\[
(x,t) \models \neg \phi \iff \neg((x,t) \models \phi)
\]

\[
(x,t) \models \phi \land \psi \iff (x,t) \models \phi \land (x,t) \models \psi
\]

\[
(x,t) \models \phi \lor \psi \iff (x,t) \models \phi \lor (x,t) \models \psi
\]

\[
(x,t) \models \phi U_{[a,b]} \psi \iff \exists t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \psi
\]

\[
\forall t_2 \in [t_1], (x,t_2) \models \phi
\]

\[
(x,t) \models F_{[a,b]} \phi \iff \exists t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \phi
\]

\[
(x,t) \models G_{[a,b]} \phi \iff \forall t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \phi
\]

The length of a formula \( h^\phi \) is introduced in Definition 3 and can be interpreted as the horizon of future predicted signals \( \mathbf{x} \) that is needed to calculate the satisfaction of a formula. For instance, if \( h^\phi = 5 \), then we need all values \( x(t') \) in the range of \( t' \in [t, t+5] \) to calculate the fulfillment of the formula.

**Definition 2.** The STL semantics are defined as [13], [10]:

\[
(x,t) \models \mu \iff f(x(t)) > 0
\]

\[
(x,t) \models \neg \mu \iff \neg((x,t) \models \mu)
\]

\[
(x,t) \models \phi \land \psi \iff (x,t) \models \phi \land (x,t) \models \psi
\]

\[
(x,t) \models \phi \lor \psi \iff (x,t) \models \phi \lor (x,t) \models \psi
\]

\[
(x,t) \models \phi U_{[a,b]} \psi \iff \exists t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \psi
\]

\[
\forall t_2 \in [t_1], (x,t_2) \models \phi
\]

\[
(x,t) \models F_{[a,b]} \phi \iff \exists t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \phi
\]

\[
(x,t) \models G_{[a,b]} \phi \iff \forall t_1 \in [t+a,t+b] \text{ s.t. } (x,t_1) \models \phi
\]

The length of a formula \( h^\phi \) is introduced in Definition 3 and can be interpreted as the horizon of future predicted signals \( \mathbf{x} \) that is needed to calculate the satisfaction of a formula. For instance, if \( h^\phi = 5 \), then we need all values \( x(t') \) in the range of \( t' \in [t, t+5] \) to calculate the fulfillment of the formula.

**Definition 3.** The formula length is inductively defined as [15]:

\[
h^\mu = 0
\]

\[
h^\mu \mu = h^\mu
\]

\[
h^\phi U_{[a,b]} \psi = b + \max(h^\phi, h^\psi)
\]

\[
h^{G_{[a,b]} \phi} = h^{F_{[a,b]} \phi} = b + h^\phi
\]

\[
h^{\phi \land \psi} = h^{\phi \lor \psi} = \max(h^\phi, h^\psi)
\]

Throughout this paper we will assume that a STL formula is in Positive Normal Form (PNF). This means that no negations (\( \neg \)) occur within a formula except of if they are placed directly in front of predicates. For instance, \( \neg G_{[a,b]} f(x(t)) \) is not in PNF whereas \( G_{[a,b]} \neg f(x(t)) \) is in PNF. This assumption does not restrict our work since every formula can be translated into PNF as explained in [15].

C. Average Space Robustness

Additionally to syntax and qualitative semantics, STL defines quantitative semantics. Space robustness \( \rho^\phi(x,t) \) is a quantitative measure that has been introduced in [13] and is
stated in Definition 4. It determines how well a formula $\phi$ is satisfied.

**Definition 4.** Space Robustness is inductively defined as [13], [16], [15]:

$$\rho^\mu(x, t) = f(x(t))$$
$$\rho^{\neg \phi}(x, t) = -\rho^\phi(x, t)$$
$$\rho^{\phi \land \psi}(x, t) = \min(\rho^\phi(x, t), \rho^\psi(x, t))$$
$$\rho^{\phi \lor \psi}(x, t) = \max(\rho^\phi(x, t), \rho^\psi(x, t))$$
$$\rho^{\mu U[a, b]}(x, t) = \max_{t_1 \in [t+a, t+b]} \min(\rho^\mu(x, t_1), \rho^\phi(x, t_2))$$
$$\rho^{F[a, b]}(x, t) = \min_{t_1 \in [t+a, t+b]} \rho^\phi(x, t_1)$$
$$\rho^{G[a, b]}(x, t) = \frac{1}{b-a} \int_{t+a}^{t+b} \rho^\phi(x, t')dt'$$

By using min-operations, space robustness considers the weakest points of a signal where $\phi$ is least satisfied. We propose a novel robustness measure $\mathcal{A}^\phi(x, t)$ in Definition 5 called Average Space Robustness (ASR), where instead average satisfaction is considered. An analogy to control synthesis with ASR is $H_2$-control as in [22], where the $H_2$ system norm is to be minimized. $H_\infty$-control minimizes the $H_\infty$ system norm and assumes the worst case scenario similar to space robustness, whereas $H_2$ uses an average approach resulting in better average performance. Another advantage is that min-operations of eventually- and until-operators are replaced by linear expressions. This makes ASR easier to use since it was already noted in [23] that analytical evaluation of space robustness is not easy due to min- and max-operations.

**Definition 5.** Average Space Robustness is inductively defined as:

$$\mathcal{A}^\mu(x, t) = f(x(t))$$
$$\mathcal{A}^{\neg \phi}(x, t) = -\mathcal{A}^\phi(x, t)$$
$$\mathcal{A}^{\phi \land \psi}(x, t) = \min(\mathcal{A}^\phi(x, t), \mathcal{A}^\psi(x, t))$$
$$\mathcal{A}^{\phi \lor \psi}(x, t) = \max(\mathcal{A}^\phi(x, t), \mathcal{A}^\psi(x, t))$$
$$\mathcal{A}^{\mu U[a, b]}(x, t) = \max_{t_1 \in [t+a, t+b]} \min(\mathcal{A}^\mu(x, t_1), \mathcal{A}^\phi(x, t_2))$$
$$\mathcal{A}^{F[a, b]}(x, t) = \min_{t_1 \in [t+a, t+b]} \mathcal{A}^\phi(x, t_1)$$
$$\mathcal{A}^{G[a, b]}(x, t) = \frac{1}{b-a} \int_{t+a}^{t+b} \mathcal{A}^\phi(x, t')dt'$$

By choosing the parameter $t_1$ beforehand it is possible to define a relaxed version of ASR in order to remove the max-operation of the always-operator. This prior determination needs to be done carefully since it affects the optimality of ASR. The simplified version is called Simplified Average Space Robustness (SASR) and defined in Definition 6.

**Definition 6.** Simplified Average Space Robustness is inductively defined as:

$$\mathcal{A}_{S}^{\mu}(x, t) = f(x(t))$$
$$\mathcal{A}_{S}^{\neg \phi}(x, t) = -\mathcal{A}_{S}^\phi(x, t)$$
$$\mathcal{A}_{S}^{\phi \land \psi}(x, t) = \min(\mathcal{A}_{S}^\phi(x, t), \mathcal{A}_{S}^\psi(x, t))$$
$$\mathcal{A}_{S}^{\phi \lor \psi}(x, t) = \max(\mathcal{A}_{S}^\phi(x, t), \mathcal{A}_{S}^\psi(x, t))$$
$$\mathcal{A}_{S}^{\mu U[a, b]}(x, t) = \frac{1}{2} \left[ \frac{1}{t_1 - t} \int_{t_1}^{t_1} \mathcal{A}_{S}^\phi(x, t')dt' + \mathcal{A}_{S}^\phi(x, t_1) \right]$$
$$\mathcal{A}_{S}^{F[a, b]}(x, t) = \max_{k_1 \in [k+a, k+b]} \mathcal{A}_{S}^\phi(x, k_1)$$
$$\mathcal{A}_{S}^{G[a, b]}(x, t) = \frac{1}{b-a + 1} \sum_{k_1 = k+a}^{k_1 + b} \mathcal{A}_{S}^\phi(x, k_1')$$

Note that the following is valid for space robustness: $\rho^\phi(x, t) > 0 \iff (x, t) \models \phi$. However, $\mathcal{A}_{S}^{\phi}(x, t) > 0 \iff (x, t) \models \phi$ is only true if we add some additional constraints. These constraints will be incorporated in the MPC framework introduced later on. It is again important to note that this definition is an average and not a worst case approach as in the space robustness framework [13].

As mentioned before, the presented controller will be discrete. Hence, we need to define discrete versions of ASR and SASR which is done in Definitions 7 and 8, respectively. Integrals are replaced by sums and $a, b, k_1 \in \mathbb{N}$, where $k_1$ is the discrete counterpart of $t_1$.

**Definition 7.** Discrete Average Space Robustness (DASR) is inductively defined as:

$$\mathcal{A}^\mu(x, k) = f(x(k))$$
$$\mathcal{A}^{\neg \phi}(x, k) = -\mathcal{A}^\phi(x, k)$$
$$\mathcal{A}^{\phi \land \psi}(x, k) = \min(\mathcal{A}^\phi(x, k), \mathcal{A}^\psi(x, k))$$
$$\mathcal{A}^{\phi \lor \psi}(x, k) = \max(\mathcal{A}^\phi(x, k), \mathcal{A}^\psi(x, k))$$
$$\mathcal{A}^{\mu U[a, b]}(x, k) = \frac{1}{2} \left[ \max_{k_1 \in [k+a, k+b]} \left( \frac{1}{k_1 - k + 1} \left( \frac{1}{k_1 - k + 1} \right) \right) \right]$$
$$\mathcal{A}^{F[a, b]}(x, k) = \max_{k_1 \in [k+a, k+b]} \mathcal{A}^\phi(x, k_1)$$
$$\mathcal{A}^{G[a, b]}(x, k) = \frac{1}{b-a + 1} \sum_{k' = k}^{k' + b} \mathcal{A}^\phi(x, k')$$
(DSASR) is inductively defined as:
\[ A_S^{δ}(x, k) = f(x(k)) \]
\[ A_S^{¬φ}(x, k) = ¬A_S^φ(x, k) \]
\[ A_S^{φ∧ψ}(x, k) = \min(A_S^φ(x, k), A_S^ψ(x, k)) \]
\[ A_S^{φ∨ψ}(x, k) = \max(A_S^φ(x, k), A_S^ψ(x, k)) \]
\[ A_S^{U(i,k)}(x, k) = \frac{1}{2} \left[ \frac{1}{k_1 - k + 1} \sum_{k'=k}^{k_1} A_S^φ(x, k') \right] + A_S^φ(x, k_1) \]
\[ A_S^{F(i,k)}(x, k) = A_S^φ(x, k_1) \]
\[ A_S^{G[i,k]}(x, k) = \frac{1}{b - a + 1} \sum_{k''=k+a}^{k+b} A_S^φ(x, k'') \]

**D. Stacked Predicate Vector**

In order to use STL together with the system description in (2), let us introduce the vector \( z(k) \) containing all linear predicate functions as
\[ z(k) = \begin{bmatrix} f_1(x(k)) \\ f_2(x(k)) \\ \vdots \\ f_G(x(k)) \end{bmatrix} = Cx(k) + c, \]
where \( C \in \mathbb{R}^{G \times n} \) and \( c \in \mathbb{R}^G \) are defined according to the specifications. Note that the mapping in (2) is an affine transformation. Also recall the difference between MITL and STL. The former only gives true or false statements, which STL also does. However, STL additionally gives a measure of how robust a formula is satisfied, meaning that higher values of \( f_i(x(k)) \), where \( i \in \{1, \cdots, G\} \), give greater satisfaction. These two features lead to an interesting interpretation since each predicate can be seen as a hyperplane. For instance, \( C = [1 \ -1] \) and \( c = 1 \) indicate the formula \( x_1 - x_2 > -1 \). The geometrical interpretation of this example is depicted in Fig. 1. The blue hyperplane separates the space into two regions. The region in the \( n \)-direction depicted in the figure is the satisfaction halfspace, whereas the other halfspace indicates negative values of the predicate. The distance of the blue hyperplane to the red or green hyperplane gives a notion of robustness.

Incorporating the solution \( x(k) \) of (2) with initial time \( k_0 \) into (4) and using the stacked input vector \( u_c \) allows to calculate the stacked predicate vector \( z_c \) for the prediction horizon \( N \) as
\[ z_c = F_c x(k_0) + H_c u_c + 1 \otimes c, \]
where \( 1 \in \mathbb{R}^N \) and
\[ z_c \begin{bmatrix} z(k_0 + 2) \\ \vdots \\ z(k_0 + N) \end{bmatrix} u_c = \begin{bmatrix} u(k_0) \\ u(k_0 + 1) \\ \vdots \\ u(k_0 + N - 1) \end{bmatrix} F_c = \begin{bmatrix} C A_d^2 \\ \vdots \\ C A_d^{N-1} B_d \end{bmatrix} \]
\[ H_c = \begin{bmatrix} C B_d \\ C A_d B_d \\ \vdots \\ C A_d^{N-2} B_d \end{bmatrix} \]

**E. Problem Statement**

The overall problem treated here is summarized as follows: Given a STL specification \( φ \) and a linear, time-invariant system as in (1), find an input sequence \( u(t) \) that yields \( (x(t), t) \models φ \) and maximizes \( A^φ(x, t) \). More specifically, we can state the problem as in Problem 1

**Problem 1.** Given a linear, time-invariant system (1), a STL formula \( φ \), an initial state \( x(t_0) \), a prediction horizon \( T_h \) and a cost function \( D \), compute
\[ \arg\min_{u_c} \int_{t_0}^{t_0 + T_h} D(A_S^φ(x, t'), u(t')) dt' \]
s.t. \( (x(t), t) \models φ \).
of how to predetermine $t_1$ so that the following problem formulation can be used:

$$
\text{argmax}_{u_c} \int_{t_0}^{t_0 + T_h} \mathcal{A}_s^{\phi}(x, t') dt' \quad (11a)
$$

s.t. $\dot{x}(t) = Ax(t) + Bu(t)$ \quad (11b)

$$
\mathcal{A}_s^{\phi}(x, t) > 0 \quad \forall t \in [t_0, t_0 + T_h], \quad (11c)
$$

For the sake of convenience, a minimization of the input is neglected. The discrete formulation of this approach is

$$
\text{argmax}_{u_c} \sum_{k'=k_0}^{k_0 + N} \mathcal{A}_s^{\phi}(x, k') \quad (12a)
$$

s.t. $z(k + 1) = CA_d x(k) + CB_d u(k) + c$ \quad (12b)

$$
\mathcal{A}_s^{\phi}(x, k') > 0 \quad \forall k' \in [k_0, k_0 + N], \quad (12c)
$$

where $N$ is the discrete prediction horizon. This paper focuses on bounded STL formulas and we assume that $N \geq h^\phi$.

A. Expressivity

To improve readability, predicates will be abbreviated by $z_i = f_i(x(t))$ with $i \in \{1, 2, \cdots, G_{\phi}\}$, where $f_i$ are affine in $x$. As previously noted, STL is inductively defined and hence there is an infinite amount of possible formulas. However, in this paper the focus is on STL formulas of the form

$$
\begin{align*}
\phi_1 &= z_1 U_{[a, b]} z_2 \\
\phi_2 &= F_{[a, b]} z_1 \\
\phi_3 &= G_{[a, b]} z_1 \\
\phi_4 &= \phi_i \land \phi_{i_2} \land \cdots \land \phi_{i_n} \quad \text{for all } i_1, \cdots, i_n \in \{1, 2, 3\} \\
\phi_5 &= \phi_i \lor \phi_{i_2} \lor \cdots \lor \phi_{i_n} \quad \text{for all } i_1, \cdots, i_n \in \{1, 2, 3\} \\
\phi_6 &= \phi_i (\lor \land) \phi_{i_2} (\lor \land) \cdots (\lor \land) \phi_{i_n} \quad \text{for all } i_1, \cdots, i_n \in \{1, 2, 3\} \\
\phi_7 &= \text{event} \implies \phi_i \quad \text{for all } i \in \{1, 2, 3, 4, 5, 6\},
\end{align*}
$$

where $z_1$ and $z_2$ are predicates. The first six formulas are all-time satisfying and $\phi_7$ is a so-called one-shot triggered by event, which can be a manual button or an event detected by online monitoring as described in [12, 24] or [25]. A greater expressivity may be possible and is the topic of future work.

B. Model Predictive Control

From now on, only the formulation in [12] will be considered due to the discrete implementation of a MPC. Furthermore, a notion of past satisfaction similar to the one introduced in [15] needs to be incorporated. Note that [15] incorporates past satisfaction into the constraints of a MPC, whereas our approach adds past satisfaction to the cost function in [12]. Therefore define the vector

$$
a_{\text{stack}} = \begin{bmatrix}
\mathcal{A}_s^{\phi}(x, k_0 - h^\phi + 1) \\
\mathcal{A}_s^{\phi}(x, k_0 - h^\phi + 2) \\
\vdots \\
\mathcal{A}_s^{\phi}(x, k_0) \\
\vdots \\
\mathcal{A}_s^{\phi}(x, k_0 + N - h^\phi)
\end{bmatrix}, \quad (13)
$$

which contains all $\mathcal{A}_{s_i}^{\phi}$ that need to be considered for the prediction horizon $N$ and the formula length $h^\phi$. Consider the MPC formulation

$$
\text{argmax}_{u_c} \sum_{i=1}^{N} a_{\text{stack}}(i) \quad (14a)
$$

s.t. $z(k + 1) = CA_d x(k) + CB_d u(k) + c$ \quad (14b)

$$
a_{\text{stack}}(i) > 0 \quad \forall i \in [1, N] \quad (14c)
$$

or equivalently

$$
\text{argmax}_{u_c} \sum_{k'=k_0-h^\phi+1}^{k_0+N} \mathcal{A}_s^{\phi}(x, k') \quad (15a)
$$

s.t. $z(k + 1) = CA_d x(k) + CB_d u(k) + c$ \quad (15b)

$$
\mathcal{A}_s^{\phi}(x, k') > 0 \quad \forall k' \in [k_0 - h^\phi + 1, k_0 - N + h^\phi]. \quad (15c)
$$

The formula length $h^\phi$ plays a role in the sense that it determines how many past and future robustness signals need to be accounted for since the prediction horizon is limited by $N$. Note that in this context the upper bound of the sum in [12] has been changed to $k_0 + N - h^\phi$, whereas the lower bound has been adjusted to $k_0 - h^\phi + 1$.

C. Temporal Operators on Predicates

Let us investigate the first three basic operators introduced in subsection III-A: 1) $\phi_1 = z_1 U_{[a, b]} z_2$ 2) $\phi_2 = F_{[a, b]} z_1$ 3) $\phi_3 = G_{[a, b]} z_1$

The operator $\phi_1$ can be formulated as

$$
\text{argmax}_{u_c} \frac{1}{2} \sum_{k'=k_0-h^\phi+1}^{k_0+N} \left[ z_2(k_1(k')) \right] + \frac{1}{k_1(k') - k_1 + 1} \sum_{k''=k_1(k')}^{k_1(k')} z_1(k'') \quad (16a)
$$

s.t. $z(k + 1) = CA_d x(k) + CB_d u(k) + c$ \quad (16b)

$$
z_1(k) > 0 \quad \forall k \in [k_0, k_1(k')] \quad (16c)
$$

$$
z_2(k_1(k')) > 0, \quad (16d)
$$

where the calculation of $k_1(k')$ will be explained in subsection III-E. This formulation can be further reduced to a linear program as

$$
\text{argmax}_{u_c} \mathbf{1}^T E \mathbf{z}_{\text{all}} \quad (17a)
$$

s.t. $z(k + 1) = CA_d x(k) + CB_d u(k) + c$ \quad (17b)

$$
z_1(k) > 0 \quad \forall k \in [k_0, k_1(k')] \quad (17c)
$$

$$
z_2(k_1(k')) > 0, \quad (17d)
$$

where $\mathbf{z}_{\text{all}}$ concatenates past predicates, which can be seen as constants, with $\mathbf{z}_c$ given in [3] as

$$
\mathbf{z}_{\text{all}} = \begin{bmatrix}
z(k_0 - h^\phi + 1) \\
z(k_0 - h^\phi + 2) \\
\vdots \\
z(k_0) \\
z_c
\end{bmatrix}. \quad (18)
$$
Note that \([16a\) and \([17a\) are equivalent. In general, the shape of \(E\) depends on the distribution of \(k_1(k')\). However, to form \(E\) note first that \(E \in \mathbb{R}^{N \times (G_\mu \cdot N)}\). Furthermore, the columns of \(E\) are associated with \(z_{a+d}\), whereas each row is associated with \(k'\) and needs to sum up to 1. This is a natural consequence of the normalization terms used in the definition of DSASR. This also applies for eventually- and always-operators since at each time each operator has the same weight within the cost function, i.e. the same importance.

For the until-operator, where \(G_\mu = 2\), \(E\) can be formed using the following step-by-step procedure:

1) Start with \(k' = k_0 - h^\phi + 1\) and set \(i = 1\).

2) Form a row vector, where the \(2(h^\phi + k_1(k'))\)-th column is set to 0.5, which corresponds to the term \(\frac{1}{2} \cdot z_2(k_1(k'))\). Furthermore, set all odd columns between the columns \(2(i-1) + 1\) and \(2(h^\phi + k_1(k'))\) to 0.5, \(\frac{1}{2} \cdot z_1(k_1(k'))\). Then concatenate this row vector with the matrix \(E\).

3) Stop if \(k' = k_0 + N - h^\phi\), otherwise increase \(k'\) by 1 and go back to step 2).

To illustrate this idea, assume that \(N = 4, h^\phi = 2\) and \(k_0 = 0\). Then assume
\[
k_1(k') = \begin{cases} 0 & \text{if } k' = \{-1, 0\} \\ 2 & \text{if } k' = \{1, 2\}. \end{cases} \tag{19}\]

Then
\[
E = \frac{1}{2} \cdot \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{20}\]

The operator \(\phi_2\) can be formulated as
\[
\text{argmax}_{u_c} \sum_{k' = k_0 - h^\phi + 1} z_1(k_1(k')) \tag{21a}\]
\[\text{s.t. } z(k + 1) = CA_d x(k) + CB_d u(k) + c \tag{21b}\]
\[z_1(k_1(k')) > 0 \tag{21c}\]

and similar to \([17a\) be reduced to a linear program
\[
\text{argmax}_{u_c} \sum_{k' = k_0 - h^\phi + 1} \min(A_{\delta}^{\phi_1}(x, k'), A_{\delta}^{\phi_1}(x, k')) \tag{22a}\]
\[\text{s.t. } z(k + 1) = CA_d x(k) + CB_d u(k) + c \tag{22b}\]
\[\lambda_{\delta}^{\phi_1}(x, k') > 0 \tag{22c}\]

Here \(G_\mu = 1\) and \(E\) is formed according to the following procedure:

1) Start with \(k' = k_0 - h^\phi + 1\).

2) Form a row vector, where the \((h^\phi + k_1(k'))\)-th column is set to 1. Set all other elements to 0 and concatenate this row vector with the matrix \(E\).

3) Stop if \(k' = k_0 + N - h^\phi\), else increase \(k'\) by 1 and go back to step 2).

Again, for \(N = 4, h^\phi = 2, k_0 = 0\) and \(k_1(k')\) as in \([19\), \(E\) amounts to
\[
E = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{23}\]

The operator \(\phi_3\) is formulated as
\[
\text{argmax}_{u_c} \sum_{k' = k_0 - h^\phi + 1} \frac{k' + b}{a + 1} \sum_{k'' = k' + a} z_1(k'') \tag{24a}\]
\[\text{s.t. } z(k + 1) = CA_d x(k) + CB_d u(k) + c \tag{24b}\]
\[z_1(k') > 0 \quad \forall k \in [k' + a, k' + b] \tag{24c}\]

and the linear program is
\[
\text{argmax}_{u_c} \sum_{k' = k_0 - h^\phi + 1} \frac{k' + b}{a + 1} \sum_{k'' = k' + a} z_1(k'') \tag{25a}\]
\[\text{s.t. } z(k + 1) = CA_d x(k) + CB_d u(k) + c \tag{25b}\]
\[z_1(k') > 0 \quad \forall k \in [k' + a, k' + b] \tag{25c}\]

with \(G_\mu = 1\) and \(E\) being formed as follows:

1) Start with \(k' = k_0 - h^\phi + 1\) and \(i = 1\).

2) Form a row vector, where all elements in \([a+i, b+i]\) are set to \(\frac{1}{b-a+i}\). Set all other elements to 0 and concatenate this row vector with the matrix \(E\).

3) Stop if \(k' = k_0 + N - h^\phi\), else increase \(k'\) and \(i\) by 1 and go back to step 2).

For instance, assume \(N = 6, k_0 = 0, a = 1, b = h^\phi = 3\). Then
\[
E = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}. \tag{26}\]

D. Boolean Connections

This section will explain how to calculate formulas of the form \(\phi_4\) and \(\phi_5\) as introduced in subsection III-A. In order to explain the methodology for formulas of the form \(\phi_4\), only a conjunction of two STL formulas as \(\phi = \phi_1 \land \phi_2\) will be considered where \(i, j \in \{1, 2, 3\}\). However, an extension to more temporal operators is possible and will be outlined. The problem can be expressed as
\[
\text{argmax}_{u_c} \sum_{k' = k_0 - h^\phi + 1} \min(A_{\delta}^{\phi_1}(x, k'), A_{\delta}^{\phi_1}(x, k')) \tag{27a}\]
\[\text{s.t. } z(k + 1) = CA_d x(k) + CB_d u(k) \tag{27b}\]
\[A_{\delta}^{\phi_1}(x, k') > 0 \quad \forall k \in [k_0 - h^\phi + 1, k_0 + N - h^\phi] \tag{27c}\]
\[A_{\delta}^{\phi_1}(x, k') > 0 \quad \forall k \in [k_0 - h^\phi + 1, k_0 + N - h^\phi]. \tag{27d}\]

Note that the cost function in \([27a\) is a sum of finite elements, which can be written as
\[
\min(A_{\delta}^{\phi_1}(x, k_0 - h^\phi + 1), A_{\delta}^{\phi_1}(x, k_0 - h^\phi + 1)) + \cdots + \min(A_{\delta}^{\phi_1}(x, k_0 + N - h^\phi), A_{\delta}^{\phi_1}(x, k_0 + N - h^\phi)). \tag{28}\]
Since (27a) is a max-min problem, the expression can be simplified by introducing an additional decision variable \( u_{x,n} \) with \( n \in \{1, \ldots, N\} \) for each sum element. First, define the vector \( \mathbf{u}_x \):

\[
\mathbf{u}_x = \begin{bmatrix}
  u_{x,1} \\
  \vdots \\
  u_{x,N} \\
  \mathbf{u}(k_0) \\
  \vdots \\
  \mathbf{u}(k_0 + N - 1)
\end{bmatrix}
\]  

and rewrite the problem as

argmax \( \sum_{i=1}^{N} u_{x,i} \)  

s.t. \( u_{x,1} \leq A_0^x (\mathbf{x}, k_0 - h^\phi + 1) \)  

\( u_{x,1} \leq A_0^x (\mathbf{x}, k_0 - h^\phi + 1) \)  

\( \vdots \)  

\( u_{x,N} \leq A_0^x (\mathbf{x}, k_0 + N - h^\phi) \)  

\( u_{x,N} \leq A_0^x (\mathbf{x}, k_0 + N - h^\phi) \)  

\( A^x_\phi (\mathbf{x}, k) > 0 \ \forall \ k \in [k_0 - h^\phi + 1, k_0 + N - h^\phi] \)  

\( A^x_\phi (\mathbf{x}, k) > 0 \ \forall \ k \in [k_0 - h^\phi + 1, k_0 + N - h^\phi] \)  

Note that (30) and (27) are equivalent. Furthermore, (30) is a linear program which can be formulated in vector notation with the cost function

argmax \( \mathbf{u}_x \) \( \mathbf{f}^T \mathbf{u}_x \),

where \( \mathbf{f} = [1; 0] \); \( 1 \) consists of \( N \) elements, whereas \( 0 \) consists of \( N \cdot m \) elements equal to zero. By manipulating the \( H_c \) matrix to

\[
H_{man} = \begin{bmatrix}
0_{(G_p,N),N} & H_c
\end{bmatrix},
\]

where \( 0_{(G_p,N),N} \) is a matrix consisting of zeros with \( G_p \cdot N \) rows and \( N \) columns. The stacked predicate vector from (3) can be reformulated as

\[
\mathbf{z}_c = F_c \mathbf{x}(k_0) + H_{man} \mathbf{u}_x + 1 \otimes \mathbf{c}.
\]

Again define \( \mathbf{z}_{all} \) as in (18) and reformulate the linear inequalities of (30b) - (30f) as

\[
Q \mathbf{u}_x \leq R \mathbf{z}_{all}.
\]

The \( Q \) matrix is given by

\[
Q = \begin{bmatrix}
1 & 0 & \cdots & 0 & \cdots & 0 \\
1 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\
0 & 0 & \cdots & 1 & \cdots & 0 \\
0 & 0 & \cdots & 1 & \cdots & 0
\end{bmatrix},
\]

whereas \( R \) depends on the structure of the temporal operators. Denote \( E_{\phi_i} \) and \( E_{\phi_j} \) as the corresponding matrices of \( \phi_i \) and \( \phi_j \), respectively. \( E_{\phi_i} \) and \( E_{\phi_j} \) need to be created as described in section III-C. However, note that for a conjunction more predicates are used than for a temporal operator on a predicate. Hence, the step-by-step procedure for \( E_{\phi_i} \) and \( E_{\phi_j} \) has to be adapted slightly. For each additional predicate, columns consisting of zeros need to be added. For instance, consider the eventually-operator that previously only used one predicate. A conjunction of two eventually-operators uses two predicates and therefore a new column needs to be added after each existing column. Finally, denoting \( E_{\phi_i}(n,:) \) as the \( n \)-th row of \( E_{\phi_i} \), \( R \) can be composed as

\[
R = \begin{bmatrix}
E_{\phi_i}(1,:) \\
\vdots \\
E_{\phi_i}(N,:) \\
E_{\phi_i}(1,:) \\
\vdots \\
E_{\phi_i}(N,:)
\end{bmatrix}.
\]

We have reformulated the original problem and can include other constraints like saturation of the input and the constraints of (30g) and (30h). This implementation is an exact implementation and an extension to more than one operator (\( \phi_i \land \phi_j \land \phi_k \land \cdots \)) can easily be handled by adding one additional constraint for each added conjunction. For instance, for \( \phi_i \land \phi_j \land \phi_k = \text{the constraints} \)

\[
u_{x,1} \leq A^{x}_{\phi_k}(\mathbf{x}, k_0 - h^\phi + 1) \]  

\( \vdots \)  

\( u_{x,N} \leq A^{x}_{\phi_k}(\mathbf{x}, k_0 + N - h^\phi) \)  

\( A^{x}_{\phi_k}(\mathbf{x}, k) > 0 \ \forall \ k \in [k_0 - h^\phi + 1, k_0 + N - h^\phi] \)  

need to be added to (30).

Disjunction formulas as \( \phi_3 \) can be handled by picking the best solution. Again, think of two temporal operators connected by a disjunction \( \phi = \phi_i \lor \phi_j \) where \( i, j \in \{1, 2, 3\} \). To approach this problem, calculate the optimal solution and the corresponding optimal input vector \( \mathbf{u}_{c,1} \) for

argmax \( u_c \sum_{k'=k_0-h^\phi+1}^{k_0+N-h^\phi} A^{x}_{\phi_i}(\mathbf{x}, k') \)  

s.t. \( \mathbf{z}(k+1) = C A_d \mathbf{x}(k) + C B_d \mathbf{u}(k) + \mathbf{c} \)  

\( A^{x}_{\phi_i}(\mathbf{x}, k) > 0 \ \forall \ k \in [k_0 - h^\phi_i + 1, k_0 + N - h^\phi_i] \)

and \( \mathbf{u}_{c,2} \) for

argmax \( u_c \sum_{k'=k_0-h^\phi_j}^{k_0+N-h^\phi_j} A^{x}_{\phi_j}(\mathbf{x}, k') \)  

s.t. \( \mathbf{z}(k+1) = C A_d \mathbf{x}(k) + C B_d \mathbf{u}(k) + \mathbf{c} \)  

\( A^{x}_{\phi_j}(\mathbf{x}, k) > 0 \ \forall \ k \in [k_0 - h^\phi_j + 1, k_0 + N - h^\phi_j] \).
The optimal state trajectories that result from the optimal inputs are denoted as $x_1^*$ and $x_2^*$. Afterwards, it is possible to calculate the satisfaction $p_1$ and $p_2$ for both subformulas

$$p_1 = \frac{1}{N} \sum_{k'=k_0-h^{\phi_1}+1}^{k_0+N-h^{\phi_1}} A_{S}^{\phi_1}(x_1^*, k') \tag{40}$$

$$p_2 = \frac{1}{N} \sum_{k'=k_0-h^{\phi_2}}^{k_0+N-h^{\phi_2}} A_{S}^{\phi_2}(x_2^*, k'). \tag{41}$$

The input corresponding to the biggest $p$-value will be applied to the system. Note that this solution is only an optimal solution if the time intervals of both subformulas are disjoint. In other words, for $P_i = \{x \in \mathbb{N}|a_i \leq x \leq b_i\}$ where $a_i$ and $b_i$ correspond to $\phi_1$, the solution is optimal if $P_i \cap P_j = \emptyset$. If the intersection of the intervals is not empty, a satisfying control input can still be found. However, the satisfaction of the formula might be lower compared to the optimal solution.

### E. Predetermine $k_1$

In this subsection, Algorithm 1 is used to calculate $k_1$ values beforehand. These values will depend on the number of until- and eventually-operators and on the shortest interval $[a, b]$ of these operators. It might turn out, if the number of operators is too high and the shortest interval is too short, that the problem is infeasible.

Let us first illustrate the general idea before we explain Algorithm 1. As an intuition of the $k_1$ calculation, assume $\phi = F_{[10,50]} z_1$ where we set $k_1 = k_0 + 50$. This results in the highest $A_{S}^{\phi}(x, k_0)$ in most cases since the system has the most time to settle and satisfy $\phi$. For more temporal operators, e.g., a conjunction $F_{[10,50]} z_1 \land F_{[10,50]} z_2$, we need to choose $k_1(k_0)$ for each subformula. In this case, $k_1(k_0) = 50$ and $k_1(k_0) = 30$ can be used for the first and second subformula, respectively. The reasoning why it is sometimes necessary to chose these $k_1$’s differently, can intuitively be seen by assuming that $z_1 = x - 2$ and $z_2 = -x + 2$ which can not be fulfilled at the same time.

Recall that it is necessary to fulfill a specification all the time. Fig. 2a shows the basic idea, where the $x$-axis displays a time line beginning from the current time $k_0$ up to the time $k_0 + N = k_0 + 70$, which corresponds to the prediction horizon. Each $y$-axis line represents a time interval, where the specification needs to be fulfilled. For the formula $F_{[10,50]} z(k)$, the line at $y = 4$ represents where the formula $F_{[10,50]} z(k+4)$ needs to be fulfilled. In other words, $z(k) > 0$ needs to be true for all times $k \in [k_0+4, k_0+54]$ as it can be seen from comparing values on the $x$-axis. By having a look at this graph, it is obvious that if $z > 0$ at $k_1 = k_0 + 50$, then $F_{[10,50]} z(k)$ is satisfied for all $k \in [k_0, k_0 + 40]$. For temporal operators without boolean connections, it should be sufficient to set $k_1$ equal to $b$, i.e. the end of the interval in order to get the optimum $k_1$ distribution. What we mean by optimal is the best possible solution to the optimization problem resulting in the highest satisfaction. This basic idea can be extended to boolean connections of two temporal operators that make use of $k_1$.

Fig. 2b shows a possible $k_1$ distribution for the specification $F_{[10,50]} z_1(k) \land F_{[10,50]} z_2(k)$. Note that the second subformula starts indexing at $y = 21$. The $k_1$ of the first subformula is $k_0 + 50$ as before, whereas the $k_1$ for the second subformula is at $k_0 + 30$. It is visible, that both specifications are fulfilled for times $k \in [k_0, k_0 + 20]$ and hence for the prediction horizon $N = 70$. A natural question is why it is not sufficient to put both $k_1$’s to the end of an interval. Therefore assume the formula $F_{[a,b]}(x_1 > 3) \land F_{[a,b]}(x_1 < 3)$, which will only be true if the $k_1$’s are chosen differently for each subformula.

Algorithm 1 gives an error due to infeasibility if $d_{min}$ is smaller than $N_{\phi}$, i.e. if the shortest interval length is smaller than the number of temporal operators. Otherwise, the equidistance of $k_1$’s between the $i$-th and the $(i-1)$-th operator is $d = \lfloor \frac{d_{min}}{N_{\phi}} \rfloor$ and each $k_1$ is periodically repeated with $d_{min} + 1$ starting from $b_{max} = max(b_1, \ldots, b_{N_{\phi}})$. Note that the notation $j = c(i) : -T : a(i)$ creates a vector, e.g. $j = 1 : 2 : 9$ creates the vector $j = [1 \ 3 \ 5 \ 7 \ 9]^{T}$.

### IV. Case Study

The system under consideration is a two tank process \cite{26} as in Fig. 3. The discretized system can be described by the

![Image of the two tank process](image-url)
input : Interval’s $[a_i, b_i]$ of temporal operators
output: Matrix containing $k$’s for each operator

/* Initialization */

$N_\phi$ = number of temporal operators;
$b_{max} = \max(b_i)$;

for $i = 1$ to $N_\phi$ do
  $len(i) = b_i - a_i$;
end

d_{min} = \min(len);

/* Check Feasibility */

if $N_\phi > d_{min}$ then
  Error: not feasible;
else
  /* Calculate distance between different $t_1$’s */
  $d = \left\lfloor \frac{d_{min}}{N_\phi} \right\rfloor$;
  /* Set period to the minimum interval */
  $T = d_{min} + 1$;
  /* Build basic interval */
  for $i = 1$ to $N_\phi$ do
    $e(i) = b_{max} - (i - 1) \cdot d$;
  end
  /* Project basic interval back and forth in time with period $T$ */
  for $i = 1$ to $N_\phi$ do
    for $j = e(i) : -T : a(i)$ do
      $t_1(i,j) = 1$;
    end
    for $j = e(i) : T : N_\phi$ do
      $t_1(i,j) = 1$;
    end
  end
end

Algorithm 1: Static $k_1$ distribution algorithm

difference equation

$$x(k + 1) = \begin{bmatrix} 0.79 & 0 \\ 0.176 & 0.0296 \end{bmatrix} x(k) + \begin{bmatrix} 0.281 \\ 0.0296 \end{bmatrix} u(k) + v(k).$$

(42)

The states correspond to the water height in each tank, $v(k)$ is additive noise and the input is constrained to $u(k) \in [0, 6]$. The formulas

1) $event \implies F_{[120,240]}(x_1 \geq 2)$
2) $(x_1 \geq 0) U_{[120,240]}(x_1 \leq 5)$
3) $F_{[120,240]}(x_1 \geq 2) \land (x_1 \leq 4) U_{[180,420]}(x_2 \leq 2.5)$

will be analyzed. The noise level is indicated by the Signal-to-Noise ratio as

$$\text{SNR}_{ib} = 10 \cdot \log_{10}\left(\frac{P_s}{P_v}\right),$$

(43)

where $P_s$ is the average signal power.

Fig. 4 shows $event \implies F_{[120,240]}(x_1 \geq 2)$, where $event$ is triggered at 120 seconds. After this initialization, the algorithm increases the water level to its maximum at 360 seconds. $(x_1 \geq 0) U_{[120,240]}(x_1 \leq 5)$ is an all-time satisfying formula and displayed in Fig. 5. The minimum points correspond to the times where $x_1 \leq 5$ is minimized. Finally, Fig. 6 shows $F_{[120,240]}(x_1 \geq 2) \land (x_1 \leq 4) U_{[180,420]}(x_2 \leq 2.5)$. Maxima correspond to maximizing $x_1 \geq 2$, whereas minima indicate minimizing $x_2 \leq 2.5$. The SNR for these examples ranges from 10.8625 to 13.6128 decibels and satisfaction is still ensured due to
the robust MPC implementation. Our MPC provides optimal robustness in the sense that it steers the state trajectory in the direction, where it has the farthest distance to the set of states not fulfilling the formula. This is different to the Mixed Integer Linear Programming (MILP) MPC implementation of [16], where an additional mechanism is needed to care for robustness as implemented in [15] or discussed in [27]. The drawback of these extensions is that MILP is already computationally demanding. All new mechanisms will require more computational power, especially when formulas, the prediction horizon or the state space get bigger.

The difference of Discrete Simplified Average Space Robustness and Space Robustness can qualitatively be discussed by having a look at the until-operator \( (x_1 \leq 4) U_{[180,420]} (x_2 \leq 2.5) \) in Fig. 6. The algorithm, using DSASR, tries to maximize \(-x_1 + 4\), which corresponds to \( x_1 \leq 4 \), all the time due to the average approach. Space robustness as in Definition 4 would not necessarily do the same and only try to elevate the satisfaction of the weakest point. For example, depending on the situation space robustness would rather try to maximize \( x_1 - 5 \) instead of \(-x_1 + 4\), whereas DSASR will maximize satisfaction of \( x_1 - 5 \) and \(-x_1 + 4\) at the same time.

V. DISCUSSION AND FUTURE WORK

This work introduced a Model Predictive Control framework, where Average Space Robustness has been incorporated into the cost function. This allows a direct maximization of the satisfaction of a Signal Temporal Logic formula. By avoiding possible non-convex cost functions, the problem has been reduced to a linear program. Simulation results have shown specification fulfilling results for a two tank process.

In the current framework, the parameter \( k_1 \) needs to be pre-determined. This could potentially be handled in a more adaptive way and is subject to future work. A further modification of SASR is also possible in this context. Another limitation is that we assume our prediction horizon to be greater than the formula length. However, the advantages are low computation times due to the linear program formulation, average satisfaction with better average performance and a resulting robustness against model uncertainties and noise.

Future work will also include an extension of the expressivity. The effect of normalizing all states to lie within the range of \([-1, 1]\) before incorporating them into predicate functions is also an area of future work besides a formal analysis of stability.

REFERENCES

[1] R. Alur and T. A. Henzinger, “Logics and models of real time: A survey,” in Real-Time: Theory in Practice, 1992.
[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, “Temporal-logic-based reactive mission and motion planning,” Robotics, IEEE Transactions on, vol. 25, no. 6, pp. 1370–1381, 2009.
[3] M. Guo and D. V. Dimarogonas, “Multi-agent plan reconfiguration under local ltl specifications,” The International Journal of Robotics Research, vol. 34, no. 2, pp. 218–235, 2015.
[4] M. Guo, K. H. Johansson, and D. V. Dimarogonas, “Revising motion planning under linear temporal logic specifications in partially known workspaces,” in *IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany, 2013.

[5] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for mobile robots,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2020–2025.

[6] S. G. Loizou and K. J. Kyriakopoulos, “Automatic synthesis of multi-agent motion tasks based on LT specifications,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1. IEEE, 2004, pp. 153–158.

[7] A. Nikou, J. Tumova, and D. V. Dimarogonas, “Cooperative task planning of multi-agent systems under timed temporal specifications,” *arXiv preprint arXiv:1509.09137*, 2015.

[8] P. Wolper, “Constructing automata from temporal logic formulas: A tutorial?” in *Lectures on Formal Methods and Performance Analysis*. Springer, 2001, pp. 261–277.

[9] C. Baier, J.-P. Katoen et al., *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[10] O. Maler, D. Nickovic, and A. Pnueli, “From MITL to timed automata,” in *Formal Modeling and Analysis of Timed Systems*. Springer, 2006, pp. 274–289.

[11] T. Henzinger, “It’s about time: Real-time logics reviewed,” *Concurrency Theory*, pp. 439–454, 1998.

[12] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[13] A. Donzé and O. Maler, “Robust satisfaction of temporal logic over real-valued signals,” in *Proceedings of the 8th international conference on Formal modeling and analysis of timed systems*. Springer-Verlag, 2010, pp. 92–106.

[14] G. E. Fainekos and G. J. Pappas, “Robustness of temporal logic specifications for continuous-time signals,” *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[15] S. Sadaraddini and C. Belta, “Robust temporal logic model predictive control,” in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 772–779.

[16] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, “Model predictive control with signal temporal logic specifications,” in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 81–87.

[17] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, “Reactive synthesis from signal temporal logic specifications,” in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 239–248.

[18] E. Camacho and C. Bordons, *Model Predictive Control*. London: Springer London, 2007.

[19] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[20] R. Findeisen and F. Allgöwer, “An introduction to nonlinear model predictive control,” in *21st Benelux Meeting on Systems and Control*, vol. 11, 2002, pp. 119–141.

[21] H. Chen and F. Allgöwer, “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability,” in *Control Conference (ECC), 1997 European*. IEEE, 1997, pp. 1421–1426.

[22] T. Glad, *Control Theory: Multivariable & Nonlinear Methods*. CRC, 2000.

[23] C. Yoo and C. Belta, “Control with probabilistic signal temporal logic,” *arXiv preprint arXiv:1510.08474*, 2015.

[24] A. Donzé, T. Ferrere, and O. Maler, “Efficient robust monitoring for STL,” in *Computer Aided Verification*. Springer, 2013, pp. 264–279.

[25] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, “Robust online monitoring of signal temporal logic,” in *Runtime Verification*. Springer, 2015, pp. 55–70.

[26] K. J. Åström and B. Wittenmark, *Computer-controlled systems*. Prentice-Hall, Inc., 1997.

[27] S. S. Farahani, V. Raman, and R. M. Murray, “Robust model predictive control for signal temporal logic synthesis,” *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, 2015.