Robust Control for Signal Temporal Logic Specifications using Discrete Average Space Robustness [*]

Lars Lindemann [a], Dimos V. Dimarogonas [a]

[a] *Department of Automatic Control, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Malvinas väg 10, SE-10044, Stockholm, Sweden*

Abstract

Control systems that satisfy temporal logic specifications have become increasingly popular due to their applicability to robotic systems. Existing control methods, however, are computationally demanding, especially when the problem size becomes too large. In this paper, a robust and computationally efficient model predictive control framework for signal temporal logic specifications is proposed. We introduce discrete average space robustness, a novel quantitative semantic for signal temporal logic, that is directly incorporated into the cost function of the model predictive controller. The optimization problem entailed in this framework can be written as a convex quadratic program when no disjunctions are considered and results in a robust satisfaction of the specification. Furthermore, we define the predicate robustness degree as a new robustness notion. Simulations of a multi-agent system subject to complex specifications demonstrate the efficacy of the proposed method.

Key words: Formal methods-based control; signal temporal logic; model predictive control; autonomous systems

1 Introduction

Formal verification and model checking with specifications in temporal logics [2] have extensively been studied during the last decade. The dynamical system under consideration is abstracted into a finite state transition system, whereas the specification is translated into a language equivalent büchi automaton. These two representations are then used to systematically check whether or not the specification is satisfied. Conversely, formal methods-based control tries to find control inputs such that the dynamical system satisfies the specification. Formal methods-based control has been investigated with linear temporal logic (LTL) [8,12,7,3] and metric interval temporal logic (MITL) [19]. LTL and MITL formulas are used to express the system specifications and can be translated into büchi automata [28,18], which allows the calculation of a product automaton with the finite state transition system. Using Dijkstra or a depth-first search algorithm, a suitable discrete run, satisfying the specification, can be found.

Signal temporal logic (STL) was introduced in [17] within the context of monitoring temporal properties of continuous-time signals for continuous and hybrid systems. The authors in [17] propose temporal property monitors that check these in STL expressed properties. For instance, for continuous and hybrid control systems it is of great interest to check whether or not the controller satisfies the control specification. STL encompasses a quantitative notion of time and space. The latter property has been used to introduce space robustness (SR) [6], a quantitative semantic stating how robustly a formula is satisfied. SR is a special case of the robust semantics in [9] for MITL where it was shown that the robust semantics are an under-approximation of the robustness degree. The robustness degree hence indicates how much a signal can be perturbed by noise before changing the truth value of the specification. Consequently, it is possible to use STL to measure how robustly and hence how well a specification is satisfied. This is in contrast to LTL where only a boolean satisfaction is given. STL was used for control by means of model predictive control (MPC) in [22] where SR is incorporated into a mixed integer linear program, which is computationally expensive. Robust extensions of this approach have been reported in [23,25,10]. Additional quantitative semantics have recently been presented. The authors in [6] not only define space robustness, but also time robustness and a combination of both. A

measure based on the weighted edit distance has been proposed in [14], while [5] equips quantitative STL semantics with the possibility to freeze operators. The connection between linear, time-invariant filtering and quantitative semantics has been made in [24]. The quantitative semantics in [1], called averaged STL, extend the semantics in [6] by so called averaged operators that average SR over certain time intervals, which naturally leads to space and time robustness. We remark that averaged STL has a similar name to what we define as discrete average space robustness; however, these semantics have fundamental differences. LTL robustness has been considered in [27].

The main motivation of this paper is the lack of computationally efficient algorithms for formal methods-based control, especially for systems with fast dynamics where computation time is a critical resource. The contributions of this paper are as follows: we first introduce discrete average space robustness (DASR) as new quantitative semantics over discrete-time signals for STL. DASR semantics are computationally more tractable than SR, but are not an under-approximation of the robustness degree; however, we account for this drawback by including additional constraints into our proposed control strategy. Second, we introduce the notion of the predicate robustness degree and show in an example that DASR, used in an optimization framework, results in a higher predicate robustness degree than SR. Third, a simplified version of DASR is directly incorporated into the cost function of a MPC framework for a fragment of STL formulas. We show that the encapsulated optimization problem can be written as a convex quadratic program. Current approaches as in [25,22,23] end up with mixed integer linear programs that become inherently nonconvex if SR is maximized. We directly maximize robustness in a computationally efficient manner. Previous results of our work have been published in [15]. This paper extends [15] by giving geometrical robustness interpretations in terms of hyperplanes and introducing the predicate robustness degree. We motivate why DASR is a good choice for control and explain the k_1 calculation for the simplified version of DASR that is missing in [15].

Section 2 introduces notation and preliminaries, while Section 3 defines discrete average space robustness. Section 4 states the formal problem, while Section 5 presents the proposed solution. Simulations are given in Section 6, followed by an outlook and conclusions in Section 7.

2 Preliminaries and Notation

Scalars are denoted by lowercase, non-bold letters x. Column vectors are lowercase, bold letters $\boldsymbol{x}$ and matrices are denoted by uppercase, non-bold letters X. The n-th row of X will be represented by $X(n,:)$ and similarly $X(:,n)$ represents the n-th column of X. The n-dimensional vector space over the real numbers $\mathbb{R}$ is $\mathbb{R}^n$,

whereas $\mathbb{N}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ are the sets of natural, non-negative, and positive real numbers, respectively. True and false are denoted by $\top$ and $\bot$, while $\otimes$ denotes the Kronecker product; $\mathbf{1}_N$ and $\mathbf{0}_N$ are column vectors containing N ones and zeros, respectively, whereas $\underline{0}_{n,n}$ denotes an $n \times n$ matrix consisting of zeros. Let $\|\boldsymbol{x}\|_M^2 := \boldsymbol{x}^T M \boldsymbol{x}$. For two sets $\mathcal{X}$ and $\mathcal{Y}$, the set-valued map $F : \mathcal{X} \rightrightarrows \mathcal{Y}$ maps each $\boldsymbol{x} \in \mathcal{X}$ to a set $F(\boldsymbol{x}) \subseteq \mathcal{Y}$.

2.1 Signals and Systems

Consider the discrete-time, linear system

$$\boldsymbol{x}(k+1) := A\boldsymbol{x}(k) + B\boldsymbol{u}(k) \qquad (1)$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$. Assume that (1) has been obtained by sampling a continuous-time system with the sampling function $\tau : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ where $\tau(0) := 0$. In this paper, periodically sampled systems are considered where $\tau(k) := kT$ with T being the sampling period. For readability reasons, the abbreviation $\tau_k := \tau(k)$ is used. For a given continuous-time interval $[a,b]$ with $a, b \in \mathbb{R}_{\geq 0}$ and $a \leq b$, define the corresponding discrete-time counterpart as the set

$$\Omega(a,b) := \{k \in \mathbb{N} | a \leq \tau(k) \leq b\}. \qquad (2)$$

Remark 1 *The definition of $\Omega(a,b)$ allows to impose continuous-time specifications on discrete-time systems.*

2.2 Signal Temporal Logic

Signal temporal logic (STL) consists of predicates μ that are obtained after evaluation of a predicate function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. In particular, for $\boldsymbol{\xi} \in \mathbb{R}^n$, let

$$\mu := \begin{cases} \top \text{ if } f(\boldsymbol{\xi}) \geq 0 \\ \bot \text{ if } f(\boldsymbol{\xi}) < 0. \end{cases} \qquad (3)$$

Let now $\boldsymbol{x} : \mathbb{N} \rightarrow \mathbb{R}^n$ be a discrete-time signal and possibly a solution to (1), then $f(\boldsymbol{x}(k))$ determines the truth value of μ at time k. Let N_μ indicate the number of predicates under consideration and let the set of predicates be $\mathcal{P} := \{\mu_1, \mu_2, \cdots, \mu_{N_\mu}\}$; $\mu_i \in \mathcal{P}$ at time k is evaluated as in (3) by $f_i(\boldsymbol{x}(k))$, which is abbreviated by

$$z_i(k) := f_i(\boldsymbol{x}(k)) \quad \forall i \in \{1, 2, \cdots, N_\mu\}$$

for readability reasons. The predicate vector $\boldsymbol{z}(k)$, gathering all predicate functions $z_i(k)$, is then defined as

$$\boldsymbol{z}(k) := \begin{bmatrix} z_1(k) & \dots & z_{N_\mu}(k) \end{bmatrix}^T. \qquad (4)$$

If, for all $i \in \{1, \dots, N_\mu\}$, $z_i(k) := f_i(\boldsymbol{x}(k))$ is affine in $\boldsymbol{x}(k)$, the predicate functions can be combined with the

system in (1) to obtain a compact linear form. Define

$$\boldsymbol{z}(k) := \begin{bmatrix} z_1(k) & \ldots & z_{N_\mu}(k) \end{bmatrix}^T := C\boldsymbol{x}(k) + \boldsymbol{c}, \quad (5)$$

where $C \in \mathbb{R}^{N_\mu \times n}$ and $\boldsymbol{c} \in \mathbb{R}^{N_\mu}$ encode the predicates under consideration. Using the prediction horizon N, let $\boldsymbol{z}_{st} := \begin{bmatrix} \boldsymbol{z}(k_0+1)^T & \boldsymbol{z}(k_0+2)^T & \ldots & \boldsymbol{z}(k_0+N)^T \end{bmatrix}^T$ and $\boldsymbol{u}_{st} := \begin{bmatrix} \boldsymbol{u}(k_0)^T & \boldsymbol{u}(k_0+1)^T & \ldots & \boldsymbol{u}(k_0+N-1)^T \end{bmatrix}^T$ to denote predicates and inputs at different times with respect to the initial time k_0, respectively. Inserting the solution $\boldsymbol{x}$ of (1) with initial time k_0 into (5), we get

$$\boldsymbol{z}_{st} := H_1 \boldsymbol{x}(k_0) + H_2 \boldsymbol{u}_{st} + \mathbf{1}_N \otimes \boldsymbol{c}, \quad (6)$$

where H_1 and H_2 follow recursively by (1) and (5).

Considering $\mu \in \mathcal{P}$, the STL syntax is now given by

$$\phi ::= \top \mid \mu \mid \neg\phi \mid \phi \wedge \psi \mid \phi\, U_{[a,b]}\, \psi ,$$

where ϕ, ψ are STL formulas. The STL semantics have been defined over continuous-time signals [17,6]. Inferring satisfaction of a formula defined over continuous-time signals by considering discrete-time signals has been addressed in [9, Section 4] and is possible under some conditions on τ and $\boldsymbol{x}$. We define the semantics for STL over discrete-time signals by minor changes in the definitions of [17,6], hence resembling the discrete definitions in [9, Section 3]. The satisfaction relation $(\boldsymbol{x}, k) \models \phi$ denotes if the signal $\boldsymbol{x}$ satisfies ϕ at time k. Note the use of $\Omega(a,b)$ from (2) and recall $\tau_k := \tau(k)$.

Definition 1 *Given $\mu_i \in \mathcal{P}$, STL formulas ϕ and ψ, and a signal $\boldsymbol{x}$, the STL semantics [6,22] are defined as:*

$$
\begin{aligned}
(\boldsymbol{x}, k) \models \mu_i \quad &\Leftrightarrow \quad f_i(\boldsymbol{x}(k)) \geq 0 \\
(\boldsymbol{x}, k) \models \neg\phi \quad &\Leftrightarrow \quad \neg((\boldsymbol{x}, k) \models \phi) \\
(\boldsymbol{x}, k) \models \phi \wedge \psi \quad &\Leftrightarrow \quad (\boldsymbol{x}, k) \models \phi \wedge (\boldsymbol{x}, k) \models \psi \\
(\boldsymbol{x}, k) \models \phi\, U_{[a,b]}\, \psi \quad &\Leftrightarrow \quad \exists k_1 \in \Omega(\tau_k + a, \tau_k + b)\ s.t. \\
&\quad (\boldsymbol{x}, k_1) \models \psi \wedge \forall k_2 \in \Omega(\tau_k, \tau_{k_1}), (\boldsymbol{x}, k_2) \models \phi
\end{aligned}
$$

Disjunction-, eventually-, and always-operator are derived as $\phi \vee \psi := \neg(\neg\phi \wedge \neg\psi)$, $F_{[a,b]}\phi := \top\, U_{[a,b]}\, \phi$, and $G_{[a,b]}\phi := \neg F_{[a,b]} \neg\phi$, respectively. The discrete length h_d^ϕ of a formula ϕ is defined in Definition 2 and can be interpreted as the prediction horizon that is sufficient to evaluate $(\boldsymbol{x}, k) \models \phi$. For instance, if $h_d^\phi := 5$, then having knowledge of $\boldsymbol{x}(k')$ for all $k' \in \{k, k+1, \ldots, k+5\}$ is sufficient to evaluate $(\boldsymbol{x}, k) \models \phi$.

Definition 2 *The continuous formula length h_c^ϕ is defined as [17]:* $h_c^\mu := 0$, $h_c^{\neg\phi} := h_c^\phi$, $h_c^{\phi\, U_{[a,b]}\, \psi} := b + \max(h_c^\phi, h_c^\psi)$, $h_c^{G_{[a,b]}\phi} := h_c^{F_{[a,b]}\phi} := b + h_c^\phi$, $h_c^{\phi\wedge\psi} := h_c^{\phi\vee\psi} := \max(h_c^\phi, h_c^\psi)$. *The discrete formula length is*

$h_d^\phi := \max_{k^* \in \Omega(0, h_c^\phi)} k^*$, *i.e., h_d^ϕ is the largest element in $\Omega(0, h_c^\phi)$.*

2.3 Robustness Degree and Space Robustness

Robustness of a discrete-time signal $\boldsymbol{x}$ with respect to a formula ϕ has been defined by the discrete-time robustness degree, robustness degree for short, as how much the signal $\boldsymbol{x}$ can be perturbed before the boolean evaluation of $(\boldsymbol{x}, k) \models \phi$ changes its truth value [9]. Let $\mathcal{L}_k(\phi) := \{\boldsymbol{x} : \mathbb{N} \to \mathbb{R}^n | (\boldsymbol{x}, k) \models \phi\}$ be the set of discrete-time signals that satisfy ϕ at time k. The closeness of two signals $\boldsymbol{x}$ and $\boldsymbol{x}^*$ is given by $\rho(\boldsymbol{x}, \boldsymbol{x}^*) := \sup_{k \in \mathbb{N}} d(\boldsymbol{x}(k), \boldsymbol{x}^*(k))$ where d is a metric assigning a distance in $\mathbb{R}^n$; d can be the Euclidean distance $d(\boldsymbol{x}(k), \boldsymbol{x}^*(k)) := \|\boldsymbol{x}(k) - \boldsymbol{x}^*(k)\|$. The distance of $\boldsymbol{x}$ to the set $\mathcal{L}_k(\phi)$ is defined as

$$\delta_k(\boldsymbol{x}, \phi) := \inf_{\boldsymbol{x}^* \in \mathrm{cl}(\mathcal{L}_k(\phi))} \rho(\boldsymbol{x}, \boldsymbol{x}^*),$$

where $\mathrm{cl}(\mathcal{L}_k(\phi))$ denotes the closure of $\mathcal{L}_k(\phi)$. The robustness degree is now given in Definition 3.

Definition 3 *Given a formula ϕ and a signal $\boldsymbol{x}$, the robustness degree at time k is defined as [9, Definition 23]:*

$$\mathcal{RD}_k(\boldsymbol{x}, \phi) := \begin{cases} -\delta_k(\boldsymbol{x}, \phi) & \text{if } \boldsymbol{x} \notin \mathcal{L}_k(\phi) \\ \delta_k(\boldsymbol{x}, \neg\phi) & \text{if } \boldsymbol{x} \in \mathcal{L}_k(\phi) \end{cases}$$

Furthermore, there are quantitative semantics that state how robustly ϕ is satisfied. Such quantitative semantics are given by space robustness (SR)[6], which reasons over continuous-time signals. A modified discrete-time version, denoted by $\rho^\phi(\boldsymbol{x}, k)$, is given in Definition 4, which resembles the discrete-time robust semantics in [9, Definition 26]. SR is an under-approximation of the robustness degree, i.e., $-\delta_k(\boldsymbol{x}, \phi) \leq \rho^\phi(\boldsymbol{x}, k) \leq \delta_k(\boldsymbol{x}, \neg\phi)$ [9, Theorem 28] so that $|\rho^\phi(\boldsymbol{x}, k)| \leq |\mathcal{RD}_k(\boldsymbol{x}, \phi)|$. Furthermore, $(\boldsymbol{x}, k) \models \phi$ if $\rho^\phi(\boldsymbol{x}, k) > 0$ [9, Theorem 30].

Definition 4 *Given $\mu_i \in \mathcal{P}$, STL formulas ϕ and ψ, and a signal $\boldsymbol{x}$, space robustness (SR) is defined as [6,22]:*

$$
\begin{aligned}
\rho^{\mu_i}(\boldsymbol{x}, k) &:= f_i(\boldsymbol{x}(k)) = z_i(k) \\
\rho^{\neg\phi}(\boldsymbol{x}, k) &:= -\rho^\phi(\boldsymbol{x}, k) \\
\rho^{\phi\wedge\psi}(\boldsymbol{x}, k) &:= \min(\rho^\phi(\boldsymbol{x}, k), \rho^\psi(\boldsymbol{x}, k)) \\
\rho^{\phi\vee\psi}(\boldsymbol{x}, k) &:= \max(\rho^\phi(\boldsymbol{x}, k), \rho^\psi(\boldsymbol{x}, k)) \\
\rho^{\phi\, U_{[a,b]}\, \psi}(\boldsymbol{x}, k) &:= \max_{k_1 \in \Omega(\tau_k+a, \tau_k+b)} \min\big(\rho^\psi(\boldsymbol{x}, k_1), \\
&\qquad\qquad \min_{k_2 \in \Omega(\tau_k, \tau_{k_1})} \rho^\phi(\boldsymbol{x}, k_2)\big) \\
\rho^{F_{[a,b]}\phi}(\boldsymbol{x}, k) &:= \max_{k_1 \in \Omega(\tau_k+a, \tau_k+b)} \rho^\phi(\boldsymbol{x}, k_1) \\
\rho^{G_{[a,b]}\phi}(\boldsymbol{x}, k) &:= \min_{k_2 \in \Omega(\tau_k+a, \tau_k+b)} \rho^\phi(\boldsymbol{x}, k_2)
\end{aligned}
$$

3

3 Discrete Average Space Robustness and the Predicate Robustness Degree

Discrete average space robustness is defined in Section 3.1, while Section 3.2 investigates how robustness is affected by changes in $\boldsymbol{z}(k)$. The results of Section 3.2 are then extended in Section 3.3 to define the predicate robustness degree. In the remainder, we consider the formula ϕ to be in positive normal form (PNF) [2, Appendix A.3]. For formulas in PNF, negations only occur in front of predicates, e.g., $\neg G_{[a,b]}\mu_1$ is not in PNF, while $G_{[a,b]}\neg\mu_1 := G_{[a,b]}\mu_2$ with $\mu_2 := \neg\mu_1$ is in PNF. A formula that is not in PNF can be rewritten in PNF [25]. Subsequently, we assume that ϕ contains no negations since each negation can be encoded into a new predicate, e.g., $\mu_2 := \neg\mu_1$ for the example above.

3.1 Discrete Average Space Robustness

Space robustness (Definition 4) uses min-operations to consider the point of weakest satisfaction within a signal. First, define $k_{min} := \min_{k^* \in \Omega(\tau_k+a,\tau_k+b)} k^*$ and $k_{max} := \max_{k^* \in \Omega(\tau_k+b,\tau_k+b)} k^*$ as the smallest and largest element of $\Omega(\tau_k + a, \tau_k + b)$, respectively. We now propose novel quantitative semantics $\mathcal{A}^\phi(\boldsymbol{x},k)$ in Definition 5, which are called discrete average space robustness (DASR) and where instead average satisfaction is considered. An advantage of DASR compared to SR is that min-operations of the always- and until-operators are replaced by linear expressions. This makes DASR computationally more tractable than SR.

Definition 5 *Given $\mu_i \in \mathcal{P}$, STL formulas ϕ and ψ, and a signal $\boldsymbol{x}$, discrete average space robustness (DASR) is defined as:*

$$\mathcal{A}^{\mu_i}(\boldsymbol{x},k) := f_i(\boldsymbol{x}(k)) = z_i(k)$$
$$\mathcal{A}^{\neg\phi}(\boldsymbol{x},k) := -\mathcal{A}^\phi(\boldsymbol{x},k)$$
$$\mathcal{A}^{\phi\wedge\psi}(\boldsymbol{x},k) := \min(\mathcal{A}^\phi(\boldsymbol{x},k),\mathcal{A}^\psi(\boldsymbol{x},k))$$
$$\mathcal{A}^{\phi\vee\psi}(\boldsymbol{x},k) := \max(\mathcal{A}^\phi(\boldsymbol{x},k),\mathcal{A}^\psi(\boldsymbol{x},k))$$
$$\mathcal{A}^{\phi\,U_{[a,b]}\,\psi}(\boldsymbol{x},k) := \max_{k_1 \in \Omega(\tau_k+a,\tau_k+b)} \frac{1}{2}\left(\frac{1}{k_1-k+1}\right.$$
$$\left.\cdot\sum_{k'=k}^{k_1}\mathcal{A}^\phi(\boldsymbol{x},k') + \mathcal{A}^\psi(\boldsymbol{x},k_1)\right)$$
$$\mathcal{A}^{F_{[a,b]}\phi}(\boldsymbol{x},k) := \max_{k_1 \in \Omega(\tau_k+a,\tau_k+b)}\mathcal{A}^\phi(\boldsymbol{x},k_1)$$
$$\mathcal{A}^{G_{[a,b]}\phi}(\boldsymbol{x},k) := \frac{1}{k_{max}-k_{min}+1}\sum_{k'=k_{min}}^{k_{max}}\mathcal{A}^\phi(\boldsymbol{x},k')$$

To further remove the max-operations of the eventually- and until-operators, k_1 can be manually chosen. A method to determine k_1 is described in Section 5.1. This results in a simplified version of DASR in Definition 6, which we call discrete simplified average space robustness (DSASR) and denote by $\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k)$.

Definition 6 *Given $\mu_i \in \mathcal{P}$, $k_1 \in \mathbb{N}$ with $k_1 \geq k$, STL formulas ϕ and ψ, and a signal $\boldsymbol{x}$, discrete simplified average space robustness (DSASR) is defined as:*

$$\mathcal{A}_\mathcal{S}^{\mu_i}(\boldsymbol{x},k) := f_i(\boldsymbol{x}(k)) = z_i(k)$$
$$\mathcal{A}_\mathcal{S}^{\neg\phi}(\boldsymbol{x},k) := -\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k)$$
$$\mathcal{A}_\mathcal{S}^{\phi\wedge\psi}(\boldsymbol{x},k) := \min(\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k),\mathcal{A}_\mathcal{S}^\psi(\boldsymbol{x},k))$$
$$\mathcal{A}_\mathcal{S}^{\phi\vee\psi}(\boldsymbol{x},k) := \max(\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k),\mathcal{A}_\mathcal{S}^\psi(\boldsymbol{x},k))$$
$$\mathcal{A}_\mathcal{S}^{\phi\,U_{[a,b]}\,\psi}(\boldsymbol{x},k) := \frac{1}{2}\left(\frac{1}{k_1-k+1}\sum_{k'=k}^{k_1}\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k')\right.$$
$$\left. + \mathcal{A}_\mathcal{S}^\psi(\boldsymbol{x},k_1)\right)$$
$$\mathcal{A}_\mathcal{S}^{F_{[a,b]}\phi}(\boldsymbol{x},k) := \mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k_1)$$
$$\mathcal{A}_\mathcal{S}^{G_{[a,b]}\phi}(\boldsymbol{x},k) := \frac{1}{k_{max}-k_{min}+1}\sum_{k'=k_{min}}^{k_{max}}\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k')$$

Note that unlike SR, neither DASR nor DSASR satisfy that $(\boldsymbol{x},k) \models \phi$ if $\mathcal{A}^\phi(\boldsymbol{x},k) > 0$ or $\mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x},k) > 0$. This can be seen by considering $\phi := G_{[a,b]}(x \geq 0)$, where it is possible that, even if $\mathcal{A}^\phi(\boldsymbol{x},k) = \frac{1}{k_{max}-k_{min}+1}\sum_{k'=k_{min}}^{k_{max}} x(k') > 0$, there might exist a $k_2 \in \Omega(\tau_k + a, \tau_k + b)$ s.t. $x(k_2) < 0$ and hence $\rho^\phi(\boldsymbol{x},k) = \min_{k_2 \in \Omega(\tau_k+a,\tau_k+b)} x(k_2) < 0$. Subsequently, $\mathcal{A}^\phi(\boldsymbol{x},k) > 0 \not\Rightarrow (\boldsymbol{x},k) \models \phi$, whereas $\rho^\phi(\boldsymbol{x},k) > 0 \Rightarrow (\boldsymbol{x},k) \models \phi$. However, in this paper additional constraints will be included in the optimization problem to enforce this property. Another property of DASR and DSASR is the following: it holds that $\mathcal{A}^{G_{[a,b]}\phi}(\boldsymbol{x},k) \neq \mathcal{A}^{\neg F_{[a,b]}\neg\phi}(\boldsymbol{x},k)$ and $\mathcal{A}_\mathcal{S}^{G_{[a,b]}\phi}(\boldsymbol{x},k) \neq \mathcal{A}_\mathcal{S}^{\neg F_{[a,b]}\neg\phi}(\boldsymbol{x},k)$ although $G_{[a,b]}\phi = \neg F_{[a,b]}\neg\phi$. Furthermore, $\mathcal{A}^{\top\mathcal{U}_{[a,b]}\phi}(\boldsymbol{x},k) \neq \mathcal{A}^{F_{[a,b]}\phi}(\boldsymbol{x},k)$ and $\mathcal{A}_\mathcal{S}^{\top\mathcal{U}_{[a,b]}\phi}(\boldsymbol{x},k) \neq \mathcal{A}_\mathcal{S}^{F_{[a,b]}\phi}(\boldsymbol{x},k)$ although $\top\mathcal{U}_{[a,b]}\neg\phi = F_{[a,b]}\phi$. These cases, however, will not occur in the STL fragment that is considered in Section 4. We remark that the goal is to obtain expressive and computationally tractable semantics.

Remark 2 *Averaged STL was introduced in [1] and is different compared to DASR and DSASR. The averaged always-, eventually-, and until-operators of averaged STL form a weighted time average over $\rho^{G_{[a,b]}\phi}(\boldsymbol{x},k)$, $\rho^{F_{[a,b]}\phi}(\boldsymbol{x},k)$, and $\rho^{\phi\,U_{[a,b]}\,\psi}(\boldsymbol{x},k)$, respectively, hence not removing min- or max-operations and keeping nonlinear, nonsmooth, and nonconvex semantics of the always-, eventually-, and until-operators. This can cause computational burdens in optimization problems. Furthermore, averaged STL results in time and space robustness, while DASR and DSASR only consider a notion of space robustness. Time robustness is a useful measure with the drawback of resulting in even more complex semantics.*

3.2 Predicate Functions as a Robustness Indicator

Define a vector $\boldsymbol{\zeta} := \begin{bmatrix} \zeta_1 & \zeta_2 & \ldots & \zeta_{N_\mu} \end{bmatrix}^T \in \mathbb{R}^{N_\mu}_{>0}$ and consider the possibly nonlinear predicate vector $\boldsymbol{z}(k)$ as in (4). Furthermore, define a modified predicate vector $\boldsymbol{z}_{\boldsymbol{\zeta}}(k) := \boldsymbol{z}(k) + \boldsymbol{\zeta}$. It hence holds that $\boldsymbol{z}(k) < \boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ where $<$ indicates element-wise inequality. Assume that the formula ϕ contains the predicates $\mathcal{P} := \{\mu_1, \ldots, \mu_{N_\mu}\}$ and that $(\boldsymbol{x}, k) \models \phi$ is evaluated with two different predicate vectors $\boldsymbol{z}(k)$ and $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$. In other words, each predicate $\mu_1, \ldots, \mu_{N_\mu}$ is evaluated as in (3) with the corresponding elements of either $\boldsymbol{z}(k)$ or $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$. According to Definitions 4, 5, and 6, we define and obtain $\rho^\phi(\boldsymbol{x}, k)$, $\mathcal{A}^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi(\boldsymbol{x}, k)$ if $\boldsymbol{z}(k)$ is used or $\rho^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, $\mathcal{A}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$ if $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ is used.

Corollary 1 *Consider a formula ϕ in PNF and two predicate vectors $\boldsymbol{z}(k)$ and $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ such that $\boldsymbol{z}(k) < \boldsymbol{z}_{\boldsymbol{\zeta}}(k)$. It then holds that $\rho^\phi(\boldsymbol{x}, k) < \rho^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, $\mathcal{A}^\phi(\boldsymbol{x}, k) < \mathcal{A}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi(\boldsymbol{x}, k) < \mathcal{A_S}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$. Furthermore, it holds that $\rho^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \rho^\phi(\boldsymbol{x}, k) \geq \zeta_{min}$, $\mathcal{A}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \mathcal{A}^\phi(\boldsymbol{x}, k) \geq \zeta_{min}$, and $\mathcal{A_S}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \mathcal{A_S}^\phi(\boldsymbol{x}, k) \geq \zeta_{min}$ where $\zeta_{min} := \min_{i \in \{1, \ldots, N_\mu\}} \zeta_i$.*

PROOF. A formula ϕ in PNF excludes the case of negations in Definitions 4, 5, and 6. Due to the inductive definition of SR, DASR, DSASR, and the exclusion of negations, a bigger predicate vector $\boldsymbol{z}(k)$ will lead to bigger functions $\rho^\phi(\boldsymbol{x}, k)$, $\mathcal{A}^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi(\boldsymbol{x}, k)$. Since $\boldsymbol{z}(k) < \boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ holds element-wise, we can conclude that $\rho^\phi(\boldsymbol{x}, k) < \rho^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, $\mathcal{A}^\phi(\boldsymbol{x}, k) < \mathcal{A}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi(\boldsymbol{x}, k) < \mathcal{A_S}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k)$. It is then straightforward to show that the difference of $\rho^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \rho^\phi(\boldsymbol{x}, k)$, $\mathcal{A}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \mathcal{A}^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}^\phi_{\boldsymbol{\zeta}}(\boldsymbol{x}, k) - \mathcal{A_S}^\phi(\boldsymbol{x}, k)$ is lower bounded by $\zeta_{min} = \min_{i \in \{1, \ldots, N_\mu\}} \zeta_i$. ∎

Next, assume that $\boldsymbol{z}(k)$ is an affine function in $\boldsymbol{x}$ as in (5), i.e., $\boldsymbol{z}(k) := C\boldsymbol{x}(k) + \boldsymbol{c}$. In Corollary 1, an increase in the predicate vector from $\boldsymbol{z}(k)$ to $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ was induced by $\boldsymbol{\zeta}$. If an increase in $\boldsymbol{z}(k)$ is instead induced by a change of $\boldsymbol{x}(k)$ in $\boldsymbol{z}(k) := C\boldsymbol{x}(k) + \boldsymbol{c}$, then robustness can be interpreted in terms of the distance between hyperplanes. Note that for each predicate function $z_i(k) := C(i,:)\boldsymbol{x}(k) + c_i$ with $i \in \{1, \ldots, N_\mu\}$, we can define a corresponding time-dependent hyperplane $\mathcal{Z}_i(k) := \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{n}_i^T \boldsymbol{x} = e_i(k)\}$ with $\boldsymbol{n}_i^T := C(i,:)$ and $e_i(k) := z_i(k) - c_i$. Now, consider two different signals $\boldsymbol{x}'$ and $\boldsymbol{x}''$ resulting in two different versions $z_i'(k) := C(i,:)\boldsymbol{x}'(k) + c_i$ and $z_i''(k) := C(i,:)\boldsymbol{x}''(k) + c_i$ of the i-th predicate function. The resulting two hyperplanes are $\mathcal{Z}_i'(k) := \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{n}_i^T \boldsymbol{x} = e_i'(k)\}$ and $\mathcal{Z}_i''(k) := \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{n}_i^T \boldsymbol{x} = e_i''(k)\}$ with $e_i'(k) := z_i'(k) - c_i$ and $e_i''(k) := z_i''(k) - c_i$. The distance between these two parallel hyperplanes at time k is given by $\delta_{\mathcal{Z}_i', \mathcal{Z}_i''}(k) := \frac{|e_i'(k) - e_i''(k)|}{\|\boldsymbol{n}_i\|} = \frac{|z_i'(k) - z_i''(k)|}{\|\boldsymbol{n}_i\|}$. Assume again that ϕ contains the predicates $\mathcal{P} := \{\mu_1, \ldots, \mu_{N_\mu}\}$ and consider the two predicate vectors

$$\boldsymbol{z}'(k) := \begin{bmatrix} z_1'(k) & z_2'(k) & \ldots & z_{N_\mu}'(k) \end{bmatrix}^T = C\boldsymbol{x}'(k) + \boldsymbol{c} \text{ and}$$

$$\boldsymbol{z}''(k) := \begin{bmatrix} z_1''(k) & z_2''(k) & \ldots & z_{N_\mu}''(k) \end{bmatrix}^T = C\boldsymbol{x}''(k) + \boldsymbol{c}.$$

Similarly to Corollary 1, define $\rho_I^\phi(\boldsymbol{x}, k)$, $\mathcal{A}_I^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}_I^\phi(\boldsymbol{x}, k)$ if $\boldsymbol{z}'(k)$ is used and $\rho_{II}^\phi(\boldsymbol{x}, k)$, $\mathcal{A}_{II}^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}_{II}^\phi(\boldsymbol{x}, k)$ if $\boldsymbol{z}''(k)$ is used in Definitions 4, 5, and 6. Next, Corollary 2 provides a geometric interpretation since the robustness increase given by $\rho_{II}^\phi(\boldsymbol{x}, k) - \rho_I^\phi(\boldsymbol{x}, k)$, $\mathcal{A}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A}_I^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A_S}_I^\phi(\boldsymbol{x}, k)$ is proportional to the minimum distance of the hyperplanes $\mathcal{Z}_i'(k^*)$ and $\mathcal{Z}_i''(k^*)$ over all $i \in \{1, \ldots, N_\mu\}$ and $k \in \{k, \ldots, k + h_d^\phi\}$. Together, Corollary 1 and 2 state that robustness increases if $\boldsymbol{z}(k)$ increases element-wise.

Corollary 2 *Consider a formula ϕ in PNF and two signals $\boldsymbol{x}'$ and $\boldsymbol{x}''$ such that $\boldsymbol{z}'(k^*) < \boldsymbol{z}''(k^*)$ for all $k^* \in \{k, \ldots, k + h_d^\phi\}$. It then holds that $\rho_I^\phi(\boldsymbol{x}, k) < \rho_{II}^\phi(\boldsymbol{x}, k)$, $\mathcal{A}_I^\phi(\boldsymbol{x}, k) < \mathcal{A}_{II}^\phi(\boldsymbol{x}, k)$, and $\mathcal{A_S}_I^\phi(\boldsymbol{x}, k) < \mathcal{A_S}_{II}^\phi(\boldsymbol{x}, k)$. Furthermore, it holds that $\rho_{II}^\phi(\boldsymbol{x}, k) - \rho_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$, $\mathcal{A}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A}_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$, and $\mathcal{A_S}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A_S}_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$ where $\delta_{min} := \min_{i \in \{1, \ldots, N_\mu\}} \min_{k^* \in \{k, \ldots, k + h_d^\phi\}} |z_i'(k^*) - z_i''(k^*)| = \min_{i \in \{1, \ldots, N_\mu\}} \min_{k^* \in \{k, \ldots, k + h_d^\phi\}} \delta_{\mathcal{Z}_i', \mathcal{Z}_i''}(k^*) \|\boldsymbol{n}_i\|$.*

PROOF. The proof is again based on the inductive definitions of SR, DASR, DSASR, and the fact that ϕ is in PNF. However, now we need to consider the minimum of $|z_i'(k^*) - z_i''(k^*)|$ over all $i \in \{1, \ldots, N_\mu\}$ and $k^* \in \{k, \ldots, k + h_d^\phi\}$ to get the lower bound δ_{min}. ∎

Example 1 *Fig. 1 displays three evaluations of the predicate function $z_1(k) := x_1(k) - x_2(k) + 1 := \boldsymbol{n}_1^T \boldsymbol{x}(k) + c_1$, which is associated with the predicate μ_1. Moving from $\boldsymbol{x}'(k) \in \mathcal{Z}_1'(k) := \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{n}_1^T \boldsymbol{x} = e_1'(k)\}$ (solid hyperplane) with $e_1'(k) := z_1'(k) - c_1 := -1$ to $\boldsymbol{x}''(k) \in \mathcal{Z}_1''(k) := \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{n}_1^T \boldsymbol{x} = e_1''(k)\}$ (dashed hyperplane) with $e_1''(k) := z_1''(k) - c_1 := 1$ results in an increase of the predicate function from $z_1'(k) = 0$ to $z_1''(k) = 2$. The distance $\delta_{\mathcal{Z}_1', \mathcal{Z}_1''}(k)$ between the solid and the dashed hyperplane is given by $\delta_{\mathcal{Z}_1', \mathcal{Z}_1''}(k) = \frac{|e_1'(k) - e_1''(k)|}{\|\boldsymbol{n}_1\|} = \sqrt{2}$ since $\|\boldsymbol{n}_1\| = \sqrt{2}$. Next, consider the formula $\phi := G_{[a,b]}\mu_1$ and assume that $|z_1'(k^*) - z_1''(k^*)| \geq 2$ for all $k^* \in \{k, \ldots, k + h_d^\phi\}$, which means that the hyperplane distance is $\delta_{\mathcal{Z}_1', \mathcal{Z}_1''}(k^*) \geq \sqrt{2}$ for all $k^* \in \{k, \ldots, k + h_d^\phi\}$. According to Corollary 2, it holds that $\rho_{II}^\phi(\boldsymbol{x}, k) - \rho_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$, $\mathcal{A}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A}_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$, and $\mathcal{A_S}_{II}^\phi(\boldsymbol{x}, k) - \mathcal{A_S}_I^\phi(\boldsymbol{x}, k) \geq \delta_{min}$ with $\delta_{min} = 2$.*
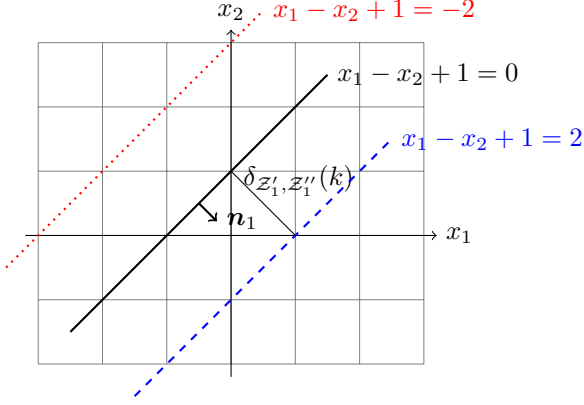
Fig. 1. Consider $z_1(k) := x_1(k) - x_2(k) + 1 := \boldsymbol{n}_1^T \boldsymbol{x}(k) + c_1$. The solid hyperplane separates the state space into two regions, one increasing and one decreasing robustness in the sense of Corollary 2. Moving the predicate $z_1(k)$ in $-\boldsymbol{n}_1$-direction from the solid to the dotted hyperplane, decreases robustness, while robustness is increased when moving into the direction of the dashed hyperplane.

3.3 Predicate Robustness Degree

Assume again nonlinear predicate functions $z_i(k) := f_i(\boldsymbol{x}(k))$ with $i \in \{1, \ldots, N_\mu\}$ as in (4). In the sequel, we will write $f_i(\boldsymbol{x}(k))$ to stress the dependence on $\boldsymbol{x}(k)$. Each predicate $\mu_i \in \mathcal{P}$ in ϕ is assigned a domain of influence at time k, denoted by $\mathcal{D}_{k,i}(\phi)$ and defined as

$$\mathcal{D}_{k,i}(\phi) := \mathcal{D}'_{k,i}(\phi) \cup \mathcal{D}'_{k,i}(\neg\phi)$$

with $\mathcal{D}'_{k,i}(\phi) := \mathbb{N} \setminus \{k' \in \mathbb{N} | \forall z \in \mathbb{R}, \forall \boldsymbol{x}^* \in f_i^{-1}(z), \exists \boldsymbol{x} \in \mathcal{L}_k(\phi) \text{ s.t. } \boldsymbol{x}(k') = \boldsymbol{x}^*\}$ where the set $f_i^{-1}(z) := \{\boldsymbol{x} \in \mathbb{R}^n | f_i(\boldsymbol{x}) = z\}$ is the inverse image of $f_i(\boldsymbol{x})$. This domain of influence states for which $k' \in \mathbb{N}$ the function values of $f_i(\boldsymbol{x}(k'))$ can change the evaluation of $(\boldsymbol{x}, k) \models \phi$. In other words, $f_i(\boldsymbol{x}(k'))$ can take arbitrary values for each $k' \notin \mathcal{D}_{k,i}(\phi)$ while the boolean evaluation of $(\boldsymbol{x}, k) \models \phi$ will always be the same. For example, consider again the formula $\phi := G_{[a,b]}\mu_1$ where $\mathcal{D}_{k,1}(\phi) = \Omega(\tau_k + a, \tau_k + b)$. In order to proceed, define the function $h(x) := \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$ and abbreviate $h_i^+(k') := h\big(f_i(\boldsymbol{x}(k'))\big)$ and $h_i^-(k') := h\big(-f_i(\boldsymbol{x}(k'))\big)$.

Definition 7 *Given a formula ϕ in PNF and a signal $\boldsymbol{x}$, the predicate robustness degree at time k is defined as*

$$\mathcal{PRD}_k(\boldsymbol{x}, \phi) :=$$
$$\begin{cases} \sum_{i=1}^{N_\mu} \sum_{k' \in \mathcal{D}_{k,i}(\phi)} h_i^-(k') f_i(\boldsymbol{x}(k')) \text{ if } \boldsymbol{x} \notin \mathcal{L}_k(\phi) \\ \sum_{i=1}^{N_\mu} \sum_{k' \in \mathcal{D}_{k,i}(\phi)} h_i^+(k') f_i(\boldsymbol{x}(k')) \text{ if } \boldsymbol{x} \in \mathcal{L}_k(\phi) \end{cases}$$

The predicate robustness degree measures robustness by considering predicates as follows: if $\boldsymbol{x} \in \mathcal{L}_k(\phi)$, we

sum over all $f_i(\boldsymbol{x}(k')) > 0$ with $i \in \{1, \ldots, N_\mu\}$ and $k' \in \mathcal{D}_{k,i}(\phi)$. If $\boldsymbol{x} \notin \mathcal{L}_k(\phi)$, we instead sum over all $f_i(\boldsymbol{x}(k')) < 0$ with $i \in \{1, \ldots, N_\mu\}$ and $k' \in \mathcal{D}_{k,i}(\phi)$. Note that if $f_i(\boldsymbol{x}(k')) \geq 0$ for all $i \in \{1, \ldots, N_\mu\}$ and $k' \in \mathcal{D}_{k,i}(\phi)$, then $(\boldsymbol{x}, k) \models \phi$ holds. Conversely, if $f_i(\boldsymbol{x}(k')) < 0$ for all $i \in \{1, \ldots, N_\mu\}$ and $k' \in \mathcal{D}_{k,i}(\phi)$, then $(\boldsymbol{x}, k) \not\models \phi$, i.e., $(\boldsymbol{x}, k) \models \neg\phi$, holds. Hence, the predicate robustness degree $\mathcal{PRD}_k(\boldsymbol{x}, \phi)$ measures the summed distance of all $f_i(\boldsymbol{x}(k')) > 0$ (if $\boldsymbol{x} \in \mathcal{L}_k(\phi)$) or $f_i(\boldsymbol{x}(k')) < 0$ (if $\boldsymbol{x} \notin \mathcal{L}_k(\phi)$) with $i \in \{1, \ldots, N_\mu\}$ and $k' \in \mathcal{D}_{k,i}(\phi)$ to $f_i(\boldsymbol{x}(k')) = 0$; $f_i(\boldsymbol{x}(k')) = 0$ is the value of the predicate where the evaluation of $(\boldsymbol{x}, k) \models \phi$ will change its truth value if all $f_i(\boldsymbol{x}(k)) > 0$ (if $\boldsymbol{x} \in \mathcal{L}_k(\phi)$) or $f_i(\boldsymbol{x}(k)) < 0$ (if $\boldsymbol{x} \notin \mathcal{L}_k(\phi)$) are changed such that $f_i(\boldsymbol{x}(k')) = 0$.

Remark 3 *The predicate robustness degree is suitable for comparing two signals $\boldsymbol{x}'$ and $\boldsymbol{x}''$ with respect to a formula ϕ. It is, however, not suited for considerations of the worst case disturbance (robustness degree).*

Example 2 *Consider the discrete-time dynamics*

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

that represent a coupled two-tank process with $T := 12$. The input u represents a pump that fills the first tank with water (water level indicated by x_1), while there is an outlet in the first tank from which water pours into the second tank (water level indicated by x_2). There is another outlet in the second tank so that water seeps into the ground. We impose the specification $\phi := G_{[144,216]}(x_1 \geq 1) \wedge G_{[372,444]}(x_1 \geq 1)$. It holds that $\mathcal{A}^\phi(\boldsymbol{x}, k) = \mathcal{A}_\mathcal{S}^\phi(\boldsymbol{x}, k)$ since the semantics for the always-operator are the same for DASR and DSASR, i.e., the notions of DASR and DSASR coincide in this example. We solve two optimization problems to demonstrate the difference between the use of $\rho^\phi(\boldsymbol{x}, k)$ (SR) and $\mathcal{A}^\phi(\boldsymbol{x}, k)$ (DASR): 1) $\underset{\boldsymbol{u}_{st}}{\operatorname{argmax}} \rho^\phi(\boldsymbol{x}, 0)$ and 2) $\underset{\boldsymbol{u}_{st}}{\operatorname{argmax}} \mathcal{A}^\phi(\boldsymbol{x}, 0)$ s.t. $x_1(k') \geq 1$ for all $k' \in \Omega(144, 216) \cup \Omega(372, 444)$. Additionally, both 1) and 2) are subject to the constraints $0 \leq u(k') \leq 3$ for all $k' \in \Omega(0, 444)$ and $\sum_{k'=0}^{h_d^\phi} u(k') \leq 20$. The result is shown in Fig. 2 and it is visible that DASR gives a greater average satisfaction within the regions $[144, 216]$ and $[372, 444]$ resulting in a greater predicate robustness degree than SR. Note that the red shaded areas show where a signal could potentially violate the formula ϕ. The predicate robustness degree at $k := 0$ is $\mathcal{PRD}_0(\boldsymbol{x}, \phi) = 5.12$ for DASR and $\mathcal{PRD}_0(\boldsymbol{x}, \phi) = 3.01$ for SR, whereas the robustness degree at $k := 0$ is $\mathcal{RD}_0(\boldsymbol{x}, \phi) = 0$ for DASR and $\mathcal{RD}_0(\boldsymbol{x}, \phi) = 0.215$ for SR. Hence, a small disturbance at exactly 216 seconds (see Fig. 2), i.e., a worst case disturbance for both shown signals, will not change $(\boldsymbol{x}, 0) \models \phi$ for the SR approach, but it can for the DASR approach. On the other hand, if there is a disturbance
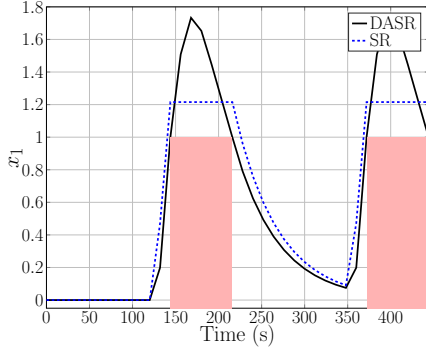
Fig. 2. For the water level x_1 of the first tank, we impose the specification $\phi := G_{[144,216]}(x_1 \geq 1) \wedge G_{[372,444]}(x_1 \geq 1)$. SR performs better in terms of the worst case disturbance, while DASR can, on average, tolerate more disturbance.

between 150 *and* 200 *seconds, the DASR approach can tolerate a bigger disturbance than the SR approach.*

One may now wonder when to choose which notion. Putting computational issues aside for a moment, it may be more suitable to use SR for safety-critical systems. For systems that are not safety-critical or where constraints are mostly soft, the notion of DASR/DSASR may be a good alternative. A worst case disturbance may now more easily lead to not satisfying the formula; however, for other disturbances the system may perform more robustly as illustrated in Example 2. Note that maximizing DASR/DSASR leads to increasing the predicate robustness degree. DASR may lead to a higher predicate robustness degree than SR. This intuitive observation can be justified by looking at the definitions of the always- and until-operator for both DASR and SR. While maximizing SR leads to maximizing the worst case, i.e., only one point in time (min-operation for until- and always-operators in Definition 4), DASR maximizes predicates over certain time intervals (summation-operator for until- and always-operators in Definition 5). These time intervals are linked to $\mathcal{D}_{k,i}(\phi)$, which are used to calculate $\mathcal{PRD}_k(\boldsymbol{x}, \phi)$.

4 Problem Statement

In this paper, the STL fragment under consideration is

$$\gamma ::= \mu \mid \neg\mu \mid \gamma_1 \wedge \gamma_2 \tag{7a}$$

$$\psi ::= \gamma_1 \, U_{[a,b]} \, \gamma_2 \mid F_{[a,b]}\gamma_1 \mid G_{[a,b]}\gamma_1 \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \tag{7b}$$

$$\phi ::= G_{[0,\infty]}\psi \mid event \implies \psi. \tag{7c}$$

where γ_1 and γ_2 are formulas of class γ given in (7a), whereas ψ_1 and ψ_2 are formulas of class ψ given in (7b). The formulas $\phi := G_{[0,\infty]}\psi$ are called *all-time satisfying formulas*, which means that the formula ψ is imposed at every sampling step; $\phi := event \implies \psi$ are called *one-time satisfying formulas* since ψ is to be satisfied once.

The boolean variable $event \in \{\top, \bot\}$ is an indicator for the time k_{event} when ψ is activated, i.e., $event(k_{event}) = \top$ and $event(k) = \bot$ if $k \in \mathbb{N} \setminus k_{event}$. This means that $(\boldsymbol{x}, 0) \models \phi$ holds if and only if $(\boldsymbol{x}, k_{event}) \models \psi$. We argue that the STL fragment in (7c) offers a good trade-off between computational tractability and expressivity and allows to express many real-world, especially robotic, specifications. Periodic tasks, such as surveillance, can be formulated as all-time satisfying formulas, while one-time satisfying formulas can be used to express reachability or other robotic tasks such as formation control. The GR(1) fragment of LTL introduced in [20] also offers computationally tractable algorithms, while considering a reactive design procedure. Compared to the GR(1) fragment, we allow for quantitative temporal requirements, i.e., hard deadlines can be imposed, while also considering until- and disjunction-operators.

Define next two variables $k_l, k_h \in \mathbb{N}$. For *all-time satisfying formulas*, set $k_l := k_0 - h_d^\psi + 1$ and $k_h := k_0 + N - h_d^\psi$ where N is a prediction horizon, k_0 is the current time, and h_d^ψ is the discrete formula length of ψ. Knowledge of $\boldsymbol{x}(k)$ for all $k \in \{k_l, k_l+1, \ldots, N\}$ is sufficient to evaluate $(\boldsymbol{x}, k') \models \psi$ for all $k' \in \{k_l, k_l + 1, \ldots, k_h\}$. This means that satisfaction of ψ is considered for the past and the future with respect to k_0, which resembles the notion of past satisfaction in [25,23]. For *one-time satisfying formulas*, set $k_l := k_h := k_{event}$ since only $(\boldsymbol{x}, k_{event}) \models \psi$ is required. We assume that the time intervals $[a, b]$ are finite and that $N \geq h_d^\psi$. To obtain computationally efficient algorithms to control the system (1) subject to an STL formula ϕ with a corresponding formula ψ as in (7), neither $\rho^\psi(\boldsymbol{x}, k)$ nor $\mathcal{A}^\psi(\boldsymbol{x}, k)$ are used and instead $\mathcal{A_S}^\psi(\boldsymbol{x}, k)$ is used. Note that the k_1's in Definition 6 are needed in order to fully define $\mathcal{A_S}^\psi(\boldsymbol{x}, k)$. Therefore, we first propose an offline algorithm, referred to as Algorithm 1, that calculates candidate k_1 for each temporal operator in ψ that then define the DSASR semantics $\mathcal{A_S}^\psi(\boldsymbol{x}, k)$. In a second step, $\mathcal{A_S}^\psi(\boldsymbol{x}, k)$ is maximized through an optimization problem, while the obtained candidate k_1's are additionally used to ensure that $(\boldsymbol{x}, k') \models \psi$ for all $k' \in \{k_l, k_l + 1, \ldots, k_h\}$ as explained more in Section 5.2.

Problem 1 *Given an STL formula ϕ as in (7c), the corresponding DSASR semantics $\mathcal{A_S}^\psi(\boldsymbol{x}, k)$ that are to be obtained by Algorithm 1, the system (1), the current state $\boldsymbol{x}(k_0)$, and a prediction horizon $N \geq h^\psi$, compute*

$$\underset{\boldsymbol{u}_{st}}{\operatorname{argmax}} \sum_{k'=k_l}^{k_h} \mathcal{A_S}^\psi(\boldsymbol{x}, k') - \sum_{k'=k_0}^{k_0+N-1} \|\boldsymbol{u}(k')\|_M^2 \tag{8a}$$

$$s.t. \; \boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k) \tag{8b}$$

$$A_{\boldsymbol{u}}\boldsymbol{u}(k') \leq \boldsymbol{b_u} \quad \forall k' \in \{k_0, \ldots, k_0 + N - 1\} \tag{8c}$$

$$A_{\boldsymbol{x}}\boldsymbol{x}(k') \leq \boldsymbol{b_x} \quad \forall k' \in \{k_0 + 1, \ldots, k_0 + N\} \tag{8d}$$

$$(\boldsymbol{x}, k') \models \psi \; \forall k' \in \Omega(\tau_{k_l}, \tau_{k_h}), \tag{8e}$$

7

where $A_{\boldsymbol{u}}$, $A_{\boldsymbol{x}}$, $\boldsymbol{b_u}$, and $\boldsymbol{b_x}$ are of appropriate size and may be used to define input and state constraints, while $M \in \mathbb{R}^{mn \times mn}$ is a positive semidefinite matrix.

In the remainder, (8b), (8c), (8d), and the term $\sum_{k'=k_0}^{k_0+N-1} \|\boldsymbol{u}(k')\|_M^2$ in (8a) are not explicitly mentioned. Solving (8) in a receding horizon fashion will result in a closed-loop solution. The procedure is initialized at time $k_0 := 0$ where (8) is solved. Only the first element of $\boldsymbol{u}_{st}$ is applied to the system and the procedure is repeated at the next sampling step with k_0 increased by 1.

Corollary 3 *The closed-loop solution obtained by iteratively solving Problem 1 results in the satisfaction of ϕ, i.e., $(\boldsymbol{x}, 0) \models \phi$, if (8) is feasible at every sampling step.*

PROOF. The assumption that (8) is feasible at every sampling step is necessary. For *one-time satisfying formulas*, the result trivially holds. For *all-time satisfying formulas*, it holds that $(\boldsymbol{x}, 0) \models \phi$ if and only if $(\boldsymbol{x}, k') \models \psi$ for all $k' \in \mathbb{N}$. Due to the choice of k_l, we consider past satisfaction and the result follows. A similar idea is used in [23] where the past $h_d^\psi - 1$ inputs and robustness constraints are stored and considered. ∎

5 Control Strategy

Section 5.1 describes the offline calculation of k_1 to obtain the DSASR semantics, while Section 5.2 explains our proposed control strategy that depends on these k_1.

5.1 Offline calculation of k_1

The parameter k_1 needs to be calculated for each eventually- and until-operator in ϕ (see Definition 6) and hence k_1^i needs to be calculated for each $i \in \{1, \ldots, N^\phi\}$, where N^ϕ is the total number of eventually- and until-operators contained in ϕ; k_1^i is therefore the k_1 of the i-th eventually- and until-operator. We first provide two examples that give some intuition on this before we present Algorithm 1 that automatically calculates k_1^i. Since Algorithm 1 uses the discrete-time intervals $\Omega(a, b)$, our examples will directly use the discrete-time intervals, e.g., $F_{\{5,\ldots,15\}}\gamma_1$ instead of $F_{[5,15]}\gamma_1$ when $\Omega(5, 15) = \{5, 6 \ldots, 15\}$. For one-time satisfying formulas, k_1^i is calculated as in the next example.

Example 3 *Assume $\phi_1 := event \implies \psi_1$ with $\psi_1 := F_{\{5,\ldots,15\}}\gamma_1$ and $N^{\phi_1} = 1$. Choosing $k_1^1 \in \{k_{event} + 5, \ldots, k_{event} + 15\}$ results in $(\boldsymbol{x}, 0) \models \phi_1$, which is the same as $(\boldsymbol{x}, k_{event}) \models \psi_1$, if $(\boldsymbol{x}, k_1^1) \models \gamma_1$ holds. If the number of eventually- and until-operators exceeds one, e.g., $N^\phi > 1$, the selection of k_1^i can be extended as follows: assume $\phi_2 := event \implies (\psi_1 \wedge \psi_2)$ with ψ_1 as before and $\psi_2 := F_{\{5,\ldots,15\}}\gamma_2$. Consequently, $N^{\phi_2} = 2$*

and k_1^i needs to be selected for $i \in \{1, 2\}$. In this case, $k_1^1 \in \{k_{event} + 5, \ldots, k_{event} + 10\}$ and $k_1^2 \in \{k_{event} + 11, \ldots, k_{event} + 15\}$ can be used for the first and second subformula, respectively. If now $(\boldsymbol{x}, k_1^1) \models \gamma_1$ and $(\boldsymbol{x}, k_1^2) \models \gamma_2$ hold, then $(\boldsymbol{x}, k_{event}) \models \psi_1 \wedge \psi_2$ holds and consequently $(\boldsymbol{x}, 0) \models \phi_2$ is true. The reason why $k_1^1 \neq k_1^2$, becomes obvious when $\gamma_1 := x \geq 2$ and $\gamma_2 := x \leq 1$.

For all-time satisfying formulas $\phi := G_{[0,\infty]}\psi$, we need $(\boldsymbol{x}, k') \models \psi$ for all $k' \in \mathbb{N}$. We hence have to find k_1^i for each $k' \in \mathbb{N}$ and each $i \in \{1, \ldots, N^\phi\}$. Therefore, we consider the functions $k_1^i : \mathbb{N} \to \mathbb{N}$ so that $k_1^i(k')$ dictates which k_1^i is used at k'.

Example 4 *Let $\phi_1 := G_{[0,\infty]}\psi_1$ with $\psi_1 := F_{\{5,\ldots,15\}}\gamma_1$ and $N^{\phi_1} = 1$. The set-valued map $K_1^1 : \mathbb{N} \rightrightarrows \mathbb{N}$ is defined by $K_1^1(k') \in \{k' + 5, k' + 15\}$. It holds that $(\boldsymbol{x}, k') \models \psi_1$ if and only if $(\boldsymbol{x}, k_1^1(k')) \models \gamma_1$ with $k_1^1(k') \in K_1^1(k')$. In Fig. 3a for instance, a vertical line at $k' = 5$, i.e., $K_1^1(5)$, indicates that γ_1 should be true at least once between $\{10, 11, \ldots, 20\}$ to obtain $(\boldsymbol{x}, 5) \models \psi_1$, i.e., if there exists $k_1^1 \in \{10, 11, \ldots, 20\}$ such that $(\boldsymbol{x}, k_1^1) \models \gamma_1$, then $(\boldsymbol{x}, 5) \models \psi_1$. By looking at Fig. 3a, it turns out that if $(\boldsymbol{x}, k_1^1(k')) \models \gamma_1$ with $k_1^1(k') := 15$ (first dotted horizontal line) for all $k' \in \{0, \ldots, 10\}$, then $(\boldsymbol{x}, k') \models \psi_1$ for all $k' \in \{0, \ldots, 10\}$. If further $(\boldsymbol{x}, k_1^1(k')) \models \gamma_1$ with $k_1^1(k') := 26$ (second dotted horizontal line) for all $k' \in \{11, \ldots, 20\}$, then $(\boldsymbol{x}, k') \models \psi_1$ for all $k' \in \{11, \ldots, 20\}$. Define the function $k_1^1(k') := k_0^1 + \Delta \lfloor \frac{k'}{\Delta} \rfloor$ with $k_0^1 := 15$ and $\Delta := 11$, where $\lfloor x \rfloor$ rounds to the nearest integer less than or equal to x. If $(\boldsymbol{x}, k_1^1(k')) \models \gamma_1$ for all $k' \in \mathbb{N}$, then $(\boldsymbol{x}, k') \models \psi_1$ for all $k' \in \mathbb{N}$ and consequently $(\boldsymbol{x}, 0) \models \phi_1$; k_0^1 is the point, referred to as baseline, which is periodically repeated with Δ. In Fig. 3a, $k_1^1(k') := k_0^1 + \Delta \lfloor \frac{k'}{\Delta} \rfloor$ corresponds to a periodic repetition of the dotted horizontal line at k_0^1 with Δ. This idea can be extended in the case of two or more eventually- and until-operators, i.e., $N^\phi > 1$. Fig. 3b shows this situation for $\phi_2 := G_{[0,\infty]}(\psi_1 \wedge \psi_2)$ with ψ_1 as before, $\psi_2 := F_{\{5,\ldots,15\}}\gamma_2$, and $N^{\phi_2} = 2$. Similarly, it is needed that $k_1^1(k') \in K_1^1(k')$ and $k_1^2(k') \in K_1^2(k')$. If $k_1^1(k') := k_0^1 + \Delta \lfloor \frac{k'+b_1-k_0^1}{\Delta} \rfloor$ with $k_0^1 := 10$ and $\Delta := 11$ is chosen for the first subformula (left subfigure) and $k_1^2(k') := k_0^2 + \Delta \lfloor \frac{k'}{\Delta} \rfloor$ with $k_0^2 := 15$ is chosen for the second subformula (right subfigure). The distance between k_0^1 and k_0^2 is given by $\eta := |k_0^1 - k_0^2| = 5$, also explained in the discussion below. If now $(\boldsymbol{x}, k_1^1(k')) \models \gamma_1$ and $(\boldsymbol{x}, k_1^2(k')) \models \gamma_2$ for all $k' \in \mathbb{N}$, then $(\boldsymbol{x}, k') \models \psi_1 \wedge \psi_2$ for all $k' \in \mathbb{N}$ and hence $(\boldsymbol{x}, 0) \models \phi$.*

Algorithm 1 next describes the procedure to calculate $k_1^i(k')$ for each $i \in \{1, \ldots, N^\phi\}$. We remark that this algorithm is not complete with respect to the feasibility of the optimization problem (8). In other words, if Algorithm 1 returns an error, it does not mean that there exists no k_1^i such that (8) is feasible. Furthermore, if Algorithm 1 returns the functions k_1^i, it does not mean that
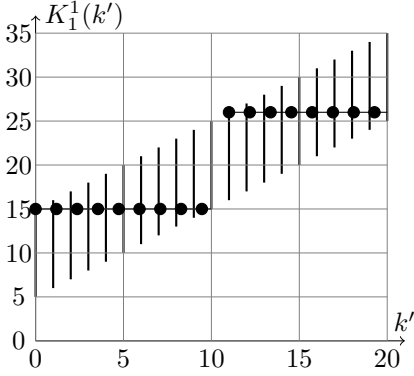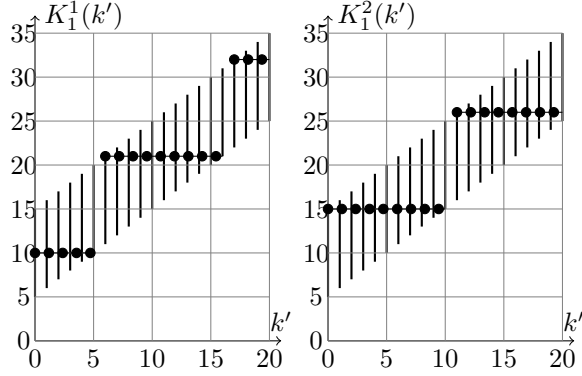
(a) Calculation of $k_1^1(k')$ for $\phi_1 = G_{[0,\infty]}\psi_1$ (b) Calculation of $k_1^1(k')$ and $k_1^2(k')$ for $\phi_2 = G_{[0,\infty]}(\psi_1 \wedge \psi_2)$

Fig. 3. The x-axis represents the discrete times $k' \in \mathbb{N}$, while the y-axis shows set-valued maps K_1^1 and K_1^2. Note that $K_1^1(k')$ and $K_1^2(k')$ map from a natural number to a set of natural numbers.

(8) is feasible. The algorithm delivers, however, candidate k_1^i that can lead to a feasible optimization problem. A lazy satisfiability modulo convex-based approach as in [26] that iterates between Algorithm 1 and (8) may account for these drawbacks. Algorithm 1 is not run on all eventually- and until-operators simultaneously, but only on a subset of them that are depending on each other. In this respect, maximal dependency clusters are defined.

Definition 8 *Consider the undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} := \{1, \ldots, N^\phi\}$ as the set of nodes. There exists an edge $(v_i, v_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ if and only if the predicates of the i-th eventually- or until-operator share at least one element of $\boldsymbol{x}$ with the j-th eventually- or until-operator; $\Xi \subseteq \mathcal{V}$ is a maximal dependency cluster if and only if $\forall v_i, v_j \in \Xi$ there is a path from v_i to v_j in $\mathcal{G}$ and $\nexists v_k \in \mathcal{V} \setminus \Xi$ such that there is a path from v_i to v_k in $\mathcal{G}$.*

Algorithm 1 is now applied to each dependency cluster $\Xi_l \in \{\Xi_1, \ldots, \Xi_{N_\Xi^\phi}\}$ separately. The inputs are Ξ_l and the intervals $[a_i, b_i]$ of each eventually- and until-operators contained in Ξ_l stacked such that $\boldsymbol{a}_{\Xi_l} := \left[a_{\Xi_l(1)}, \ldots, a_{\Xi_l(|\Xi_l|)}\right]^T$ and $\boldsymbol{b}_{\Xi_l} := \left[b_{\Xi_l(1)}, \ldots, b_{\Xi_l(|\Xi_l|)}\right]^T$ (line 1), where $\Xi_l(j)$ denotes the j-th element of Ξ_l and $|\Xi_l|$ denotes the cardinality of Ξ_l. The outputs are Δ_{Ξ_l} and k_0^i (line 19), which can then be used to determine the function k_1^i given by

$$k_1^i(k') := k_0^i + \Delta_{\Xi_l} \left\lfloor \frac{k' + h_d^{b_i} - k_0^i}{\Delta_{\Xi_l}} \right\rfloor, \qquad (9)$$

where $h_d^{b_i} := \max_{k \in \Omega(0, b_i)} k$, i.e., the discrete formula length of the i-th eventually- or until-operator. Note again that k_0^i is a baseline that is periodically repeated with Δ_{Ξ_l} as discussed in Example 4 (horizontal lines in Fig. 3). Lines 2-7 determine $d_{\Xi_l}^{min}$, i.e., the shortest discrete-time interval among all eventually- and until-operators in Ξ_l. The period $\Delta_{\Xi_l} := d_{\Xi_l}^{min} + 1$ is calculated in line 8. If the number of operators $|\Xi_l|$ is too high and the period Δ_{Ξ_l} is too short, then the algorithm returns an error (lines 9-10). Otherwise, define the distance between two consecutive $k_0^{\Xi_l(j)}$ and $k_0^{\Xi_l(j+1)}$ to be $\eta_{\Xi_l} := \lfloor \frac{\Delta_{\Xi_l}}{|\Xi_l|} \rfloor$ (line 12). It follows that $|\Xi_l|\eta_{\Xi_l} \leq \Delta_{\Xi_l}$ and consequently each eventually- and until-operator obtains a separate slot within the period Δ_{Ξ_l}, which guarantees that $k_1^i \neq k_1^j$ for all $i, j \in \Xi_l$ with $i \neq j$. In line 13, the baseline k_0^i for $i = \Xi_l(1)$ is calculated as $k_0^{\Xi_l(1)} := \min_{j \in \{1, \ldots, |\Xi_l|\}} \min_{k \in \Omega(\boldsymbol{a}_{\Xi_l}(j), \boldsymbol{b}_{\Xi_l}(j))} k$. All k_0^i with $i \in \Xi_l \setminus \Xi_l(1)$ are then calculated in lines 14-16. Note that for Example 2 we get $k_0^1 := 10$ and $k_0^2 := 15$, while Algorithm 1 outputs $k_0^1 := 5$ and $k_0^2 := 10$. Both solutions are valid, but the values in Example 4 have been chosen this way for illustrative reasons.

Algorithm 1 Calculation of Δ_{Ξ_l} and k_0^i for all $i \in \Xi_l$

1: **procedure** $k_1(\Xi_l, \boldsymbol{a}_{\Xi_l}, \boldsymbol{b}_{\Xi_l})$
2:     **for** $j := 1 : |\Xi_l|$ **do**
3:        $\boldsymbol{k}_{min}(j) := \min_{k \in \Omega(\boldsymbol{a}_{\Xi_l}(j), \boldsymbol{b}_{\Xi_l}(j))} k$
4:        $\boldsymbol{k}_{max}(j) := \max_{k \in \Omega(\boldsymbol{a}_{\Xi_l}(j), \boldsymbol{b}_{\Xi_l}(j))} k$
5:        $\boldsymbol{d}(j) := \boldsymbol{k}_{max}(j) - \boldsymbol{k}_{min}(j)$
6:     **end for**
7:     $d_{\Xi_l}^{min} := \min_{j \in \{1, \ldots, |\Xi_l|\}} \boldsymbol{d}(j)$
8:     $\Delta_{\Xi_l} := d_{\Xi_l}^{min} + 1$
9:     **if** $|\Xi_l| > \Delta_{\Xi_l}$ **then**
10:        error()   ▷ Not feasible
11:     **else**
12:        $\eta_{\Xi_l} := \lfloor \frac{\Delta_{\Xi_l}}{|\Xi_l|} \rfloor$
13:        $k_0^{\Xi_l(1)} := \min_{j \in \{1, \ldots, |\Xi_l|\}} \boldsymbol{k}_{min}(j)$
14:        **for** $j := 2 : |\Xi_l|$ **do**
15:           $k_0^{\Xi_l(j)} := k_0^{\Xi_l(1)} + (j-1)\eta_{\Xi_l}$
16:        **end for**
17:     **end if**
18:     **return** Δ_{Ξ_l}, k_0^i for all $i \in \Xi_l$
19: **end procedure**

9

5.2 Computationally-efficient Solutions via Convex Quadratic Programming

It is next shown how Algorithm 1 is used in our control approach defined in Problem 1. We emphasize beforehand that Algorithm 1 is used to obtain $\mathcal{A}_{\mathcal{S}}^{\psi}(\boldsymbol{x}, k)$ in (8a) and to make sure that (8e) holds. For all-time satisfying formulas, we show in a first step (Step A) how $G_{[0,\infty]}\psi$ with ψ as 1) $\gamma_1\, U_{[a,b]}\, \gamma_2$, 2) $F_{[a,b]}\gamma_1$, and 3) $G_{[a,b]}\gamma_1$ can be handled if γ_1 and γ_2 do not contain conjunctions. If γ_1 and/or γ_2 contain a conjunction (last case in (7a)), e.g., $\gamma_1 := \mu_1 \wedge \mu_2$ and $\gamma_2 := \mu_3 \wedge \mu_4$, then it holds that $\gamma_1\, U_{[a,b]}\, \gamma_2 = \mu_1\, U_{[a,b]}\, \mu_3 \wedge \mu_2\, U_{[a,b]}\, \mu_4$, $F_{[a,b]}\gamma_1 = F_{[a,b]}\mu_1 \wedge F_{[a,b]}\mu_2$, and $G_{[a,b]}\gamma_1 = G_{[a,b]}\mu_1 \wedge G_{[a,b]}\mu_2$ if k_1, as determined by Algorithm 1, remains the same for the until- and eventually-operator on the right-hand side of the equations (see Definition 1). This case hence reduces to the conjunction case to be discussed in the second step (Step B) where we consider $G_{[0,\infty]}\psi$ with ψ as $\psi_1 \wedge \psi_2$. Note that one-time satisfying formulas are a subclass of all-time satisfying formulas with $k_l = k_h = k_{event}$.

Step A) First, consider $\psi := \gamma_1\, U_{[a,b]}\, \gamma_2$ with $\mathcal{A}_{\mathcal{S}}^{\gamma_1}(\boldsymbol{x}, k) := z_1(k)$ and $\mathcal{A}_{\mathcal{S}}^{\gamma_2}(\boldsymbol{x}, k) := z_2(k)$. For $\phi := G_{[0,\infty]}\psi$, the equations (8a) and (8e) in Problem 1 become

$$\operatorname*{argmax}_{\boldsymbol{u}_{st}} \frac{1}{2} \sum_{k'=k_l}^{k_h} z_2\big(k_1(k')\big) + \frac{1}{k_1(k') - k' + 1} \sum_{k''=k'}^{k_1(k')} z_1(k'') \tag{10a}$$

$$\text{s.t. } z_1(k'') \geq 0 \quad \forall\, k' \in \Omega(\tau_{k_l}, \tau_{k_h}), \forall k'' \in \Omega(\tau_{k'}, \tau_{k_1(k')}) \tag{10b}$$

$$z_2\big(k_1(k')\big) \geq 0 \quad \forall\, k' \in \Omega(\tau_{k_l}, \tau_{k_h}). \tag{10c}$$

Note that the constraints (10b) and (10c) are added to account for the constraint (8e), i.e., these constraints establish the desired under-approximation property and hence lead to satisfaction of ψ at each k'. The function $k_1(k') := k_1^1(k')$ is given by (9). Second, consider $\psi := F_{[a,b]}\gamma_1$ with $\mathcal{A}_{\mathcal{S}}^{\gamma_1}(\boldsymbol{x}, k) := z_1(k)$. For $\phi := G_{[0,\infty]}\psi$, the equations (8a) and (8e) become

$$\operatorname*{argmax}_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} z_1\big(k_1(k')\big) \tag{11a}$$

$$\text{s.t. } z_1\big(k_1(k')\big) \geq 0 \quad \forall\, k' \in \Omega(\tau_{k_l}, \tau_{k_h}) \tag{11b}$$

where the constraint (11b) enforces (8e) in Problem 1. Third, consider $\psi := G_{[a,b]}\gamma_1$ with $\mathcal{A}_{\mathcal{S}}^{\gamma_1}(\boldsymbol{x}, k) := z_1(k)$. For $\phi := G_{[0,\infty]}\psi$, the equations (8a) and (8e) become

$$\operatorname*{argmax}_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} \frac{1}{k_{max,k'} - k_{min,k'} + 1} \sum_{k''=k_{min,k'}}^{k_{max,k'}} z_1(k'') \tag{12a}$$

$$\text{s.t. } z_1(k'') \geq 0 \quad \forall\, k' \in \Omega(\tau_{k_l}, \tau_{k_h}), \quad \forall\, k'' \in \Omega(\tau_{k'} + a, \tau_{k'} + b) \tag{12b}$$

where $k_{min,k'} := \min_{k^* \in \Omega(\tau_{k'}+a, \tau_{k'}+b)} k^*$, $k_{max,k'} := \max_{k^* \in \Omega(\tau_{k'}+a, \tau_{k'}+b)} k^*$, and the constraint (12b) is again added to enforce (8e) in Problem 1.

Step B) Consider $\psi := \psi_i \wedge \psi_j$ where ψ_i, ψ_j are of class ψ as in (7b) and do not contain conjunctions/disjunctions. For $\phi := G_{[0,\infty]}\psi$, the equations (8a) and (8e) become

$$\operatorname*{argmax}_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} \min(\mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x}, k'), \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x}, k')) \tag{13a}$$

$$\text{s.t. } c_{temp}^{\psi_i} \text{ and } c_{temp}^{\psi_j}, \tag{13b}$$

where $c_{temp}^{\psi_i}$ is a shorthand notation for the constraints (10b) and (10c) if $\psi_i = \gamma_1\, U_{[a,b]}\, \gamma_2$, (11b) if $\psi_i = F_{[a,b]}\gamma_1$ and (12b) if $\psi_i = G_{[a,b]}\gamma_1$.

Theorem 1 *Assume that $\boldsymbol{z}(k)$ is linear as in (5), ψ does not contain disjunctions, and that (10), (11), (12), and (13) together with Algorithm 1 are used to encode (8). Then the optimization problem (8) can be written as a convex quadratic program. If $M := \underline{0}_{m,m}$, then the optimization problem can be written as a linear program.*

PROOF. Equations (10), (11), (12) can be reduced to a convex quadratic program as $\operatorname*{argmax}_{\boldsymbol{u}_{st}} \mathbf{1}_N^T E\, \boldsymbol{z}_{all}$ s.t. $R\boldsymbol{z}_{all} \geq \mathbf{0}$ where $\boldsymbol{z}_{all}$ stacks $\boldsymbol{z}_{st}$ with past predicates as $\boldsymbol{z}_{all} := \left[\boldsymbol{z}(k_l)^T \ \ldots \ \boldsymbol{z}(k_0)^T \ \boldsymbol{z}_{st}^T\right]^T$. Note that $\boldsymbol{z}_{all}$ is linear in $\boldsymbol{u}_{st}$ as can be seen from (6) and that a linear program is obtained when $M := \underline{0}_{m,m}$. For translating (10) into E and R, note that E depends on $k_1(k')$ and is of size $E \in \mathbb{R}^{N \times N_\mu(N + h_d^\psi)}$ where $N_\mu = 1$. By denoting $i_k := i + k_l - 1$, the E matrix can be constructed as

$$E(i, j) := \begin{cases} \frac{1}{2} & \text{if: } j = 2(h_d^\psi + k_1(i_k) - k_0) \\ \frac{1}{2(k_1(i_k) - i_k + 1)} & \text{if: } 1)\, j \text{ is odd, and} \\ & \quad 2)\, 2(i-1) \leq j, \text{ and} \\ & \quad 3)\, j \leq 2(h_d^\psi + k_1(i_k) - k_0) \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Let N_c denote the total number of inequalities in (10b) and (10c). Then the matrix $R \in \mathbb{R}^{N_c \times N_\mu(N + h_d^\psi)}$ is selected such that $R(i, j)\boldsymbol{z}_{all} \geq \mathbf{0}_{N_c}$ represents all inequalities in (10b) and (10c). Next, note that (11) and (12) can be translated into E with $N_\mu = 1$, while R can be determined in the same way as for $\psi := \gamma_1\, U_{[a,b]}\, \gamma_2$. The

matrix E for (11) is then

$$E(i,j) := \begin{cases} 1 & \text{if } j = h_d^\psi + k_1(i_k) - k_0 \\ 0 & \text{otherwise,} \end{cases} \quad (15)$$

while the matrix E for (12) is

$$E(i,j) := \begin{cases} \frac{1}{\kappa} & \text{if: 1) } j \le k_{max,i_k} - k_l + 1, \text{ and} \\ & \quad 2) \ k_{min,i_k} - k_l + 1 \le j \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

where $\kappa := k_{max,i_k} - k_{min,i_k} + 1$. For conjunctions in (13), note that $\operatorname*{argmax}_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} \min(\mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x},k'), \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x},k'))$ in (13a) is a sum of finite elements as

$$\operatorname*{argmax}_{\boldsymbol{u}_{st}} \{ \min(\mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x},k_l), \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x},k_l)) + \ldots +$$
$$\min(\mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x},k_h), \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x},k_h)) \}. \quad (17)$$

With $\boldsymbol{u}_{\boldsymbol{x}} := \begin{bmatrix} u_{\boldsymbol{x},1} & \ldots & u_{\boldsymbol{x},N} & \boldsymbol{u}_{st}^T \end{bmatrix}^T$ where $u_{\boldsymbol{x},1}, \ldots, u_{\boldsymbol{x},N}$ are new decision variables, (13) can hence be written as

$$\operatorname*{argmax}_{\boldsymbol{u}_{\boldsymbol{x}}} \sum_{i=1}^{N} u_{\boldsymbol{x},i} \quad (18a)$$

$$\text{s.t. } u_{\boldsymbol{x},1} \le \mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x},k_l) \text{ and } u_{\boldsymbol{x},1} \le \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x},k_l) \quad (18b)$$

$$\vdots \quad (18c)$$

$$u_{\boldsymbol{x},N} \le \mathcal{A}_{\mathcal{S}}^{\psi_i}(\boldsymbol{x},k_h) \text{ and } u_{\boldsymbol{x},N} \le \mathcal{A}_{\mathcal{S}}^{\psi_j}(\boldsymbol{x},k_h) \quad (18d)$$

$$c_{temp}^{\psi_i} \text{ and } c_{temp}^{\psi_j}. \quad (18e)$$

Note that (18) and (13) are equivalent [4, pp. 6-7]. This is again a convex quadratic program with cost function $\operatorname*{argmax}_{\boldsymbol{u}_{\boldsymbol{x}}} \begin{bmatrix} \mathbf{1}_N^T & \mathbf{0}_{Nm}^T \end{bmatrix} \boldsymbol{u}_{\boldsymbol{x}}$. By defining $H_{2,man} := \begin{bmatrix} \mathbf{0}_{N_\mu N, N} & H_2 \end{bmatrix}$, the stacked predicate vector from (6) can be rewritten as $\boldsymbol{z}_{st} := H_1 \boldsymbol{x}(k_0) + H_{2,man} \boldsymbol{u}_{\boldsymbol{x}} + \mathbf{1}_N \otimes \boldsymbol{c}$ so that the linear inequalities of (18b) - (18d) are equivalent to $Q\boldsymbol{u}_{\boldsymbol{x}} \le R\boldsymbol{z}_{all}$, where Q and R are constructed accordingly. Explicit construction rules for Q and R are omitted due to space limitation. Note, however, that the E matrices constructed in (14), (15), and (16) can be used to form R. An extension to more than one conjunction $\psi := \psi_i \wedge \psi_j \wedge \psi_k \wedge \cdots$ can be handled by adding additional constraints for each added conjunction. ∎

Convex quadratic programs can be solved reliably and efficiently. We therefore propose in Step C) to solve disjunctions by calculating multiple convex quadratic programs in parallel instead of using computationally demanding mixed integer programs to resolve the disjunction operator, e.g., by using the Big-M method.

Step C) Consider next $\psi := \psi_i \vee \psi_j$ where ψ_i and ψ_j are of class ψ as in (7b) and do not contain any other disjunctions for now. For $\phi := G_{[0,\infty)}\psi$, we separately calculate the optimal solution of $G_{[0,\infty)}\psi_i$ and $G_{[0,\infty)}\psi_j$ with k_l and k_h based on ψ. Hence, calculate $C_\pi := \max_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} \mathcal{A}_{\mathcal{S}}^{\psi_\pi}(\boldsymbol{x},k')$ s.t. $c_{temp}^{\psi_\pi}$ for each $\pi \in \{i,j\}$ where $c_{temp}^{\psi_\pi}$ denotes the corresponding set of constraints obtained from the previous steps. These two processes are run simultaneously to then choose the corresponding input with the highest cost to be applied to the system. In other words, let $\pi^* := \operatorname{argmax}_{\pi \in \{i,j\}} C_\pi$ and apply the corresponding input $\boldsymbol{u}_{st}^*$ to (1) where $\boldsymbol{u}_{st}^*$ is calculated as $\boldsymbol{u}_{st}^* := \operatorname*{argmax}_{\boldsymbol{u}_{st}} \sum_{k'=k_l}^{k_h} \mathcal{A}_{\mathcal{S}}^{\psi_{\pi^*}}(\boldsymbol{x},k')$ s.t. $c_{temp}^{\psi_{\pi^*}}$. This procedure can be applied in exactly the same way to solve formulas ψ involving more than one disjunction, where additional disjunctions lead to additional C_π's. Due to the choice of k_l and k_h based on ψ, a receding horizon procedure will guarantee satisfaction of ϕ in the sense of Corollary 3. It could be argued that this way of handling disjunctions is rather conservative since we assume that either ψ_i or ψ_j holds for each $k' \in \{k_l, \ldots, k_h\}$. There may indeed be cases where $(\boldsymbol{x},k') \models \psi_i$ for some k' and $(\boldsymbol{x},k') \models \psi_j$ for the remaining k' yield a feasible result. Our approach, however, is computationally more favorable than solving mixed integer programs as will be further argued in the simulation results of Section 6.

We now analyze how the framework derived above can be modified when the optimization problem is infeasible, i.e., when there is no solution to (8). The idea is similar to [25] and [11] and makes use of slack variables $\zeta_i \ge 0$ with $i \in \{1, \ldots, N_\mu\}$ that are added as decision variables to the optimization problem. Therefore, recall that $\boldsymbol{\zeta} := \begin{bmatrix} \zeta_1 & \zeta_2 & \ldots & \zeta_{N_\mu} \end{bmatrix}^T$ and define $\boldsymbol{u}_{sl} := \begin{bmatrix} \boldsymbol{u}_{st}^T & \boldsymbol{\zeta}^T \end{bmatrix}^T$. Next, change the cost function in (8a) to

$$\operatorname*{argmax}_{\boldsymbol{u}_{sl}} \sum_{k'=k_l}^{k_h} \mathcal{A}_{\mathcal{S}_{\boldsymbol{\zeta}}}^{\psi}(\boldsymbol{x},k') - s\mathbf{1}_{N_\mu}^T \boldsymbol{\zeta}, \quad (19)$$

where s is a sufficiently large constant. Recall the difference between $\mathcal{A}_{\mathcal{S}_{\boldsymbol{\zeta}}}^{\psi}(\boldsymbol{x},k')$ and $\mathcal{A}_{\mathcal{S}}^{\psi}(\boldsymbol{x},k')$ as discussed in Section 3, i.e., the former is evaluated with $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ and the latter is evaluated with $\boldsymbol{z}(k)$. This leads to the fact that the constraints in (10b), (10c), (11b), and (12b) are changed to $z_1(k'') + \zeta_1 \ge 0$, $z_2(k_1(k')) + \zeta_2 \ge 0$, $z_1(k_1(k')) + \zeta_1 \ge 0$, and $z_1(k'') + \zeta_1 \ge 0$, respectively.

Corollary 4 *If the optimization problem* (8) *is infeasible, then the modified cost function in* (19) *together with the use of $\boldsymbol{z}_{\boldsymbol{\zeta}}(k)$ leads to a least violating solution of the optimization problem in* (8).

PROOF. Note that formulas ψ in (7b) are in PNF. Then, according to Corollary 1, we have that

$\mathcal{A_S}^\psi(\boldsymbol{x},k) < \mathcal{A_S}^\psi_\zeta(\boldsymbol{x},k)$ and $\mathcal{A_S}^\psi_\zeta(\boldsymbol{x},k) - \mathcal{A_S}^\psi(\boldsymbol{x},k) \geq \zeta_{min}$ where $\zeta_{min} := \min_{i\in\{1,\ldots,N_\mu\}} \zeta_i$ if $\zeta_i > 0$ for all $i \in \{1,\ldots,N_\mu\}$. The cost function minimizes ζ_i, but will select $\zeta_i > 0$ if the original problem ($\boldsymbol{\zeta} = \boldsymbol{0}_{N_\mu}$) is infeasible in order to artificially increase the robustness $\mathcal{A_S}^\psi_\zeta(\boldsymbol{x},k)$. This will eventually lead to a feasible problem, but now using $\mathcal{A_S}^\psi_\zeta(\boldsymbol{x},k)$ instead of $\mathcal{A_S}^\psi(\boldsymbol{x},k)$. The minimization of $s\boldsymbol{1}_{N_\mu}^T\boldsymbol{\zeta}$ leads to a least violating solution of the optimization problem in (8). Note that the hyperplane interpretation of Section 3 applies due to the assumption of linear predicate functions. ∎

Remark 4 *Corollary 4 does not guarantee that $(\boldsymbol{x},k) \models \psi$ for all $k \in \Omega(\tau_{k_l}, \tau_{k_h})$. We remark that [11] uses a similar idea for the specific case of predicate repair. In contrast to our approach, [25] uses a scalar slack variable ξ and defines $\boldsymbol{z}_{soft}(k) = \boldsymbol{z}(k) + \boldsymbol{1}_{N_\mu}\zeta$. This works well for SR, but for DSASR a vector $\boldsymbol{\zeta}$ is more suitable since it avoids an unnecessary increase in DSASR. For instance, consider $x_1(k) = x_2(k) := 0$ for all $k \in \mathbb{N}$. The formula $\phi := (x_1 \geq 5)\,U_{[5,10]}\,(x_1 \geq 0)$ with $\boldsymbol{z}(k) := \left[x_1(k) - 5 \; x_2(k)\right]^T$ is initially infeasible due to $x_1(k) < 5$, in fact we have $\rho^\phi(\boldsymbol{x},0) = -5$ and $\mathcal{A_S}^\phi(\boldsymbol{x},0) = -5$. If instead $\boldsymbol{z}_{soft}(k) := \boldsymbol{z}(k) + \boldsymbol{1}_{N_\mu}\zeta = \left[x_1(k) - 5 \; x_2(k)\right]^T + \left[5 \; 5\right]^T$ is used, the problem is feasible with $\rho^\phi_{soft}(\boldsymbol{x},k) = 0$ and $\mathcal{A_S}^\phi_{soft}(\boldsymbol{x},k) = 2.5$. For $\boldsymbol{z}_\zeta(k) := \boldsymbol{z}(k) + \boldsymbol{\zeta} = \left[x_1(k) - 5 \; x_2(k)\right]^T + \left[5 \; 0\right]^T$ we get $\rho^\phi_\zeta(\boldsymbol{x},k) = 0$, while $\mathcal{A_S}^\phi_\zeta(\boldsymbol{x},k) = 0$, i.e., $\mathcal{A_S}^\phi_\zeta(\boldsymbol{x},k)$ does not unnecessarily increase and indicate too optimistic robustness values.*

6 Simulations

The system under consideration is a centralized multi-agent system with three agents α_1, α_2, and α_3. Each agent α_i with $i \in \{1,2,3\}$ has four states x^i, y^i, v_x^i, and v_y^i denoting position and velocity in two-dimensional space, respectively. The agents obey continuous-time double integrator dynamics and the stacked vector of all agents is denoted by $\boldsymbol{x} := \left[\boldsymbol{x}^1 \; \boldsymbol{x}^2 \; \boldsymbol{x}^3\right]^T \in \mathbb{R}^{12}$ with $\boldsymbol{x}^i := \left[x^i \; v_x^i \; y^i \; v_y^i\right]^T \in \mathbb{R}^4$. After periodic sampling with $T := 0.1$, the discrete-time dynamics are

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k), \tag{20}$$

where $A := \mathcal{I}_6 \otimes \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}$, $B := \mathcal{I}_6 \otimes \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$, and $\boldsymbol{u}(k) := \left[u_x^1 \; u_y^1 \; u_x^2 \; u_y^2 \; u_x^3 \; u_y^3\right]^T \in [-20,20]^6$ with $\mathcal{I}_6$ being the 6×6 identity matrix. In order to obtain linear predicates,

we use the infinity norm. For instance, $\left\|\left[x^i \; y^i\right]\right\|_\infty := \max(|x^i|, |y^i|)$ can be used to rewrite $\left\|\left[x^i \; y^i\right]\right\|_\infty \leq 5$ as $(x^i \leq 5) \wedge (-x^i \leq 5) \wedge (y^i \leq 5) \wedge (-y^i \leq 5)$, i.e., the formula $\left\|\left[x^i \; y^i\right]\right\|_\infty \leq 5$ can be rewritten by using three conjunctions and four predicates with the linear predicate functions $f_1(\boldsymbol{x}) := 5 - x^i$, $f_2(\boldsymbol{x}) := 5 + x^i$, $f_3(\boldsymbol{x}) := 5 - y^i$, and $f_4(\boldsymbol{x}) := 5 + y^i$.

The specification imposed on the system, in words, is the following: each agent α_i is supposed to remain in the region $[0,10] \times [0,10]$, i.e., $\left[x^i \; y^i\right]^T \in [0,10] \times [0,10]$, and the multi-agent system is supposed to perform surveillance of this region. In formulas, the all-time satisfying formula $\phi := G_{[0,\infty)}(\psi_1 \wedge \psi_2)$ is imposed where
$$\psi_1 := G_{[0,3]}\left(\left\|\left[x^1 \; y^1 \; x^2 \; y^2 \; x^3 \; y^3\right]^T - \boldsymbol{5}_6\right\|_\infty \leq 5\right)$$
and $\psi_2 := F_{[1,3]}\left(\left\|\left[x^1 \; y^1\right]^T - \left[5 \; 9\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^1 \; y^1\right]^T - \left[1 \; 5\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^2 \; y^2\right]^T - \left[8 \; 8\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^2 \; y^2\right]^T - \left[2 \; 2\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^3 \; y^3\right]^T - \left[9 \; 5\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^3 \; y^3\right]^T - \left[5 \; 1\right]^T\right\|_\infty \leq 1\right)$; ψ_1 defines the requirement to stay within $[0,10] \times [0,10]$, while ψ_2 expresses the surveillance task that is to be performed every 1 to 3 seconds. This formulation results in 36 predicates with 35 conjunctions, which can be seen as a complex temporal logic formula.

For the simulation, we have selected a prediction horizon of $N := 45$, while no cost penalization of inputs has been used, i.e., $M := \underline{0}_{6,6}$. Within the MPC procedure, we performed 100 optimization steps. The closed-loop solution of this MPC procedure is shown in Fig. 4. The agents remain within $[0,10] \times [0,10]$ all the time, while satisfying the surveillance task within the time bounds. This would result in $(\boldsymbol{x},0) \models \phi$ by repeating the MPC procedure infinitely many times. It can also be seen that the DSASR semantics have been maximized. For instance, looking at the signal x^1 and y^1, it can be seen that $x^1(2) = 1$, $x^1(3) = 5$, $x^1(4.1) = 1.073$, $x^1(5.1) = 4.944$, $x^1(6.2) = 1.187$, and $x^1(7.2) = 4.835$, while $y^1(2) = 5$, $y^1(3) = 9$, $y^1(4.1) = 5.141$, $y^1(5.1) = 8.707$, $y^1(6.2) = 5.217$, and $y^1(7.2) = 8.81$. This shows that the reachability task $F_{[1,3]}\left(\left\|\left[x^1 \; y^1\right]^T - \left[5 \; 9\right]^T\right\|_\infty \leq 1\right) \wedge F_{[1,3]}\left(\left\|\left[x^1 \; y^1\right]^T - \left[1 \; 5\right]^T\right\|_\infty \leq 1\right)$ for agent α_1 is performed with high robustness. In fact, the robustness degree is $\mathcal{RD}_0(\boldsymbol{x},\phi) = 0.324$. We compared this result with the case where we do not maximize the DSASR semantics, i.e., not accounting for (8a) while still ensuring (8e), which results in a marginal ro-
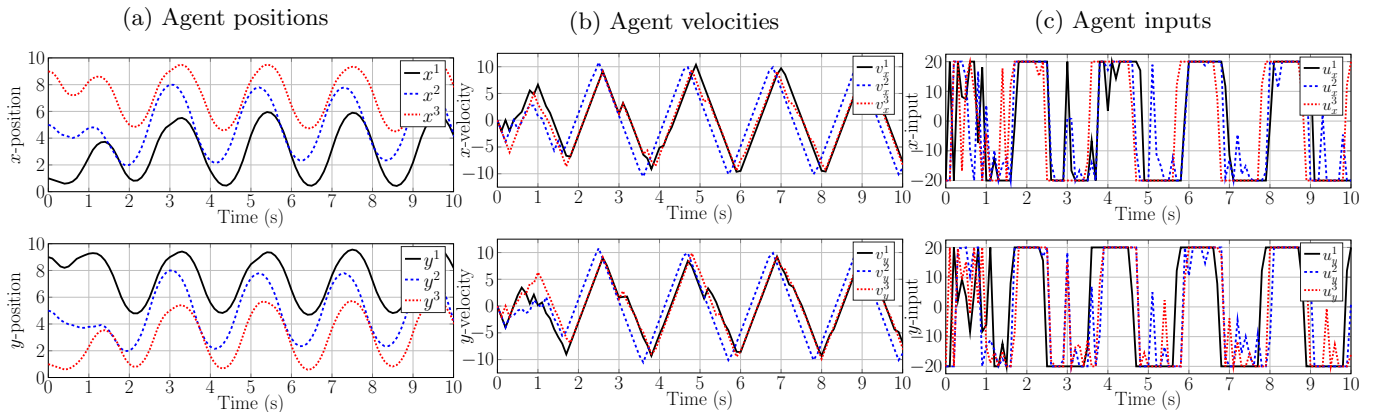
Fig. 4. Simulations result for $\phi := G_{[0,\infty)}(\psi_1 \wedge \psi_2)$.

bustness degree of only $\mathcal{RD}_0(\boldsymbol{x}, \phi) = 0.005$. Furthermore, we remark that the open-loop solution, which is not presented due to space limitations, gives optimal robustness, i.e., $x^1(3) = x^1(5.1) = x^1(7.2) = 5$, $x^1(2) = x^1(4.1) = x^1(6.2) = 1$ and $y^1(3) = y^1(5.1) = y^1(7.2) = 9$, $y^1(2) = y^1(4.1) = y^1(6.2) = 5$.

The simulation has been performed in real-time on a MacBook Air (Mid 2012) with a two-core 1.8 GHz CPU and 4 GB of RAM. The resulting linear program has 315 decision variables and, on average, 2264.3 constraints; it has been modeled in Yalmip [16] and solved by the commercial solver Gurobi [13]. The average time to solve (8) (where $\psi := \psi_1 \wedge \psi_2$) is 0.076 seconds. For comparison, we tried to solve the same formula ϕ with the same dynamics (20) by using the toolbox BluSTL [21], which is based on the works in [22,23]; however, after 30 minutes of calculations, the software still did not return a result. This confirms what the authors in [22,23] already described, i.e., the MILP formulation becomes intractable when the problem size is large. In fact, checking the feasibility of the obtained MILP formulation is NP-hard. For the robustness-based encoding, [23, Section 2.3] states the complexity in terms of variables and constraints: $O(NN_\mu) + O(N|\phi|)$ continuous variables and $O(N|\phi|)$ binary variables are introduced, where $|\phi|$ is the number of operators in ϕ. We remark that our simulation is made with low performance hardware, which indicates that our algorithms will also run efficiently on integrated microcontrollers.

7 Discussion and Future Work

This work introduced discrete average space robustness as quantitative semantics for signal temporal logic. Closely connected to these semantics, the predicate robustness degree was introduced as a new robustness notion of a signal with respect to a signal temporal logic formula by only looking at predicates. In an illustrative example, we motivated why these notions may be a suitable choice in control applications. A model

predictive control framework was then presented where a simplified version of discrete average space robustness was incorporated into the cost function of the optimization problem. The presented framework guarantees the satisfaction of the formula while maximizing the discrete average space robustness. Furthermore, it was shown that the proposed framework is computationally-efficient, which was demonstrated in a high-dimensional simulation example.

For the simplified version of discrete average space robustness, a parameter k_1 needs to be determined. We proposed an algorithm that solves this problem; however, this algorithm has some limitations and is hence subject to future work. Another topic of future work is the question if recursive feasibility of the MPC framework can be guaranteed. Furthermore, experiments are planned in order to demonstrate the suitability of our method for practical use. In a complementary approach, we are currently also working on continuous-time feedback control laws for signal temporal logic specifications by considering nonlinear predicates and nonlinear (multi-agent) systems.

References

[1] T. Akazaki and I. Hasuo. Time robustness in mtl and expressivity in hybrid system falsification. In *Proceedings of the International Conference on Computer Aided Verification*, pages 356–374, San Francisco, CA, 2015.

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking.* The MIT Press, Cambridge, MA, 1 edition, 2008.

[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14(1):61–70, 2007.

[4] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, New York, NY, 1 edition, 2004.

[5] L. Brim, T. Vejpustek, D. Šafránek, and J. Fabriková. Robustness analysis for value-freezing signal temporal logic. In *Proceedings Second International Workshop on Hybrid Systems and Biology*, pages 20–36, Taormina, Italy, 2013.

[6] 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*, pages 92–106, Klosterneuburg, Austria, 2010.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[8] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2020–2025, Barcelona, Spain, 2005.

[9] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[10] S. S. Farahani, V. Raman, and R. M. Murray. Robust model predictive control for signal temporal logic synthesis. In *Proceedings of the 5th Analysis and Design of Hybrid Systems ADHS*, pages 323 – 328, Atlanta, GA, 2015.

[11] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia. Diagnosis and repair for synthesis from signal temporal logic specifications. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 31–40, Vienna, Austria, 2016.

[12] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[13] Inc. Gurobi Optimization. Gurobi optimizer reference manual. http://www.gurobi.com, 2016.

[14] S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković. Quantitative monitoring of stl with edit distance. In *Proceedings of the 16th International Conference on Runtime Verification*, pages 201–218, Madrid, Spain, 2016.

[15] L. Lindemann and D. V. Dimarogonas. Robust motion planning employing signal temporal logic. In *Proceedings of the 2017 American Control Conference*, pages 2950–2955, Seattle, WA, 2017.

[16] J. Löfberg. Yalmip: A toolbox for modeling and optimization in matlab. In *in Proceeding of the IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, Taipei, Taiwan, 2004.

[17] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of the International Conference on FORMATS-FTRTFT*, pages 152–166, Grenoble, France, 2004.

[18] O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *Proceedings of the 4th International Conference on FORMATS*, pages 274–289, Paris, France, 2006.

[19] A. Nikou, J. Tumova, and D. V. Dimarogonas. Cooperative task planning of multi-agent systems under timed temporal specifications. In *Proceedings of the 2016 American Control Conference*, Boston, MA, 2016.

[20] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *Proceedings of the International Workshop on VMCAI*, pages 364–380, Charleston, SC, 2006.

[21] V. Raman and A. Donzé. Blustl: A model predictive control toolbox with signal temporal logic constraints. In *https://github.com/BluSTL/BluSTL*, 2016.

[22] 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 *Proceedings of the IEEE 53rd Conference on Decision and Control*, pages 81–87, Los Angeles, CA, 2014.

[23] 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*, pages 239–248, Seattle, WA, 2015.

[24] A. Rodionova, E. Bartocci, D. Nickovic, and R. Grosu. Temporal logic as filtering. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 11–20, Vienna, Austria, 2016.

[25] S. Sadraddini and C. Belta. Robust temporal logic model predictive control. In *Proceedings of the 53rd Conference on Communication, Control, and Computing*, pages 772–779, Monticello, IL, 2015.

[26] Yasser Shoukry, Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. Smc: satisfiability modulo convex optimization. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 19–28, Pittsburgh, PA, april 2017. ACM.

[27] P. Tabuada and D. Neider. Robust linear temporal logic. In *Proceedings of the 25th Conference on Computer Science Logic*, pages 10:1–10:21, Marseille, France, 2016.

[28] P. Wolper. Constructing automata from temporal logic formulas: A tutorial? In *Proceedings of Lectures on Formal Methods and Performance Analysis*, pages 261–277, Berg en Dal, The Netherlands, 2000.