Automatica 139 (2022) 110184

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Formal synthesis of closed-form sampled-data controllers for nonlinear continuous-time systems under STL specifications ☆

Cees Ferdinand Verdier [a], Niklas Kochdumper [c], Matthias Althoff [c], Manuel Mazo Jr. [b,*]

[a] Hardt Hyperloop, Paardenmarkt 1, 2611 PA, Delft, The Netherlands
[b] Delft Center of Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands
[c] Department of Informatics, Technical University of Munich, 85748 Garching, Germany

ARTICLE INFO

Article history:
Received 7 June 2020
Received in revised form 28 October 2021
Accepted 18 December 2021
Available online 19 February 2022

Keywords:
Achievable controller performance
Optimal controller synthesis for systems with uncertainties
Formal controller synthesis
Temporal logic
Reachability analysis

ABSTRACT

We propose a counterexample-guided inductive synthesis framework for the formal synthesis of closed-form sampled-data controllers for nonlinear systems to meet STL specifications over finite-time trajectories. Rather than stating the STL specification for a single initial condition, we consider an (infinite and bounded) set of initial conditions. Candidate solutions are proposed using genetic programming, which evolves controllers based on a finite number of simulations. Subsequently, the best candidate is verified using reachability analysis; if the candidate solution does not satisfy the specification, an initial condition violating the specification is extracted as a counterexample. Based on this counterexample, candidate solutions are refined until eventually a solution is found (or a user-specified number of iterations is met). The resulting sampled-data controller is expressed as a closed-form expression, enabling both interpretability and the implementation in embedded hardware with limited memory and computation power. The effectiveness of our approach is demonstrated for multiple systems.

1. Introduction

Recent years have seen a surge in interest in controller synthesis for temporal logic specifications, realizing complex behavior beyond traditional stability requirements, see, e.g., the recent literature survey in Belta and Sadraddini (2019). Originally stemming from the field of computer science, temporal logic has been used to describe the correctness of complex behaviors of computer systems (Baier & Katoen, 2008). As it originally dealt with finite systems, (bi-)simulation approaches have been proposed to abstract infinite systems to finite systems (Belta, Yordanov, & Gol, 2017; Tabuada, 2009). However, as a downside, these approaches (e.g., Girard, Pola, & Tabuada, 2010; Habets, Collins, & van Schuppen, 2006; Liu, Ozay, Topcu, & Murray, 2013; Reissig, Weber, & Rungger, 2017) typically suffer from the curse of dimensionality and return controllers in the form of enormous lookup tables (Zapreev, Verdier, & Mazo, 2018).

Where certain temporal logics, such as linear temporal logic, reason over traces of finite systems, signal temporal logic (STL) reasons over continuous signals (Maler & Nickovic, 2004). Besides a Boolean answer to whether the formula is satisfied, quantitative semantics of STL have been introduced (Donzé & Maler, 2010; Fainekos & Pappas, 2009), providing a quantitative measure on how robustly a formula is satisfied. These robustness measures enable optimization-based methods for temporal logic, such as model predictive control (MPC) (Farahani, Raman, & Murray, 2015; Lindemann & Dimarogonas, 2017; Raman, Donzé, Sadigh, Murray, & Seshia, 2015; Sadraddini & Belta, 2015; Sadraddini & Belta, 2018), optimal trajectory planning (Pant, Abbas, Quaye, & Mangharam, 2018), reinforcement learning (Aksaray, Jones, Kong, Schwager, & Belta, 2016), and neural networks (Liu, Mehdipour, & Belta, 2021; Yaghoubi & Fainekos, 2019). Apart from optimization-based methods, other proposed approaches for STL specifications rely on control barrier functions (CBF) (Garg & Panagou, 2019; Lindemann & Dimarogonas, 2019a). While the work in Lindemann and Dimarogonas (2019a) does not optimize a robustness measure of the STL specification, the computation of the control input for every time step relies on online quadratic optimization. Alternatively, in Lindemann, Verginis, and Dimarogonas (2017) and Lindemann and Dimarogonas (2019b) the synthesis for a fragment of STL is reformulated to a prescribed performance control problem, resulting in a continuous state feedback control law.

☆ This work is supported by NWO Domain TTW under the CADUSY project #13852, the ERC Starting Grant SENTIENT (755953) and the ERC Consolidator Grant justITSELF (817629). The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Necmiye Ozay under the direction of Editor Daniel Liberzon.

* Corresponding author.
E-mail addresses: cees@hardt.global (C.F. Verdier), niklas.kochdumper@tum.de (N. Kochdumper), althoff@tum.de (M. Althoff), m.mazo@tudelft.nl (M. Mazo Jr.).

https://doi.org/10.1016/j.automatica.2022.110184

While (bi-)simulation approaches provide feedback strategies for all (admissible) initial conditions, only a limited number of optimization-based approaches consider a set of initial conditions (Belta & Sadraddini, 2019), including Farahani et al. (2015), Raman et al. (2015), Schürmann, Kochdumper, and Althoff (2018). In Sadraddini and Belta (2018), tube MPC is used, in which a tube around a nominal initial condition is found for which the robustness measure is guaranteed. Similarly, the control barrier functions in Lindemann and Dimarogonas (2019a) provide a forward invariant set around the initial condition.

In this work, we utilize genetic programming (GP) (Koza, 1992) and reachability analysis (Althoff, 2010) to synthesize controllers. The benefit of genetic programming is that it is able to automatically find a structure for the controller, as the right structure is typically unknown beforehand (Belta & Sadraddini, 2019). Moreover, the resulting controllers can be verified using off-the-shelf verification methods and are generally easier to interpret than e.g. neural network controllers or look-up tables in the form of binary decision diagrams (BDDs). Genetic programming has been used for formal synthesis for reach-avoid problems in Verdier and Mazo (2018) and Verdier and Mazo (2020), in which controllers and Lyapunov-like functions are automatically synthesized for nonlinear and hybrid systems. Also, reachability analysis has been used in formal controller synthesis for reach-avoid problems, e.g., in Schürmann et al. (2018), MPC is combined with reachability analysis, whereas Ding, Li, Huang, and Tomlin (2011), Schürmann and Althoff (2017a), Schürmann, Vignali, Prandini, and Althoff (2020) synthesize a sequence of optimal control inputs (Ding et al., 2011) or linear controllers (Schürmann & Althoff, 2017a, 2017b; Schürmann et al., 2020) for a sequence of time intervals.

To the best of our knowledge, there are no closed-form controller synthesis methods which guarantee general STL specifications for a set of initial conditions. The goal of this work is to synthesize correct-by-construction closed-form controllers for nonlinear continuous-time systems subject to bounded disturbances for finite-time STL specifications. Moreover, we consider a sampled-data implementation of the controller, i.e., the controller output is only updated periodically and is held constant between sampling times. To this end, we propose a framework based on counterexample-guided inductive synthesis (CEGIS) (see e.g. Abate et al., 2017; Raman et al., 2015; Ravanbakhsh & Sankaranarayanan, 2015; Solar-Lezama, Tancau, Bodik, Seshia, & Saraswat, 2006), combining model checking for STL (Roehm, Oehlerking, Heinz, & Althoff, 2016), counterexample generation using reachability analysis (Kochdumper, Schürmann, & Althoff, 2020), and genetic programming (GP) (Koza, 1992). Our CEGIS approach combines a learning step with a formal verification step, in this case GP and reachability analysis, respectively. Violations obtained during verification are used to improve the learning process, until a controller which formally satisfies the desired specification is found, or a user-defined maximum of iterations is reached. The synthesis of a closed-form sampled-data controller for general STL specifications is NP-complete. Unsurprisingly, the proposed method is not a complete method, i.e. existence of the solution does not guarantee that a solution will be returned in a finite number of iterations. Moreover, as the method relies on simulations, reachability analysis, and SMT solvers, the (offline) computational complexity of the proposed method is significant. However, costly computations are performed offline and the method results in an interpretable closed-form sampled-data controller, enabling digital implementation with small online computational costs and a small memory footprint.

The main contributions of this work are twofold: first of all, we propose a CEGIS framework combining genetic programming with reachability analysis for the synthesis of closed-form sampled-data controllers for STL specifications. To enable reasoning over reachable sets as opposed to singular trajectories, Roehm et al. (2016) introduced reachset temporal logic (RTL) and proposed a sound transformation from STL to RTL. Our second contribution is the definition of quantitative semantics for RTL, and proving that the quantitative semantics is sound and complete. Similar to the quantitative semantics of STL, these quantitative semantics provide a measure of how robustly a formula is satisfied.

2. Preliminaries

The set of real positive numbers is denoted by $\mathbb{R}_{\geq 0}$. The interior and power set of a set S are denoted by $int(S)$ and 2^S, respectively. Finally, an n-dimensional zero vector is denoted by $\mathbf{0}_n$.

2.1. Signal temporal logic

We consider specifications expressed in signal temporal logic (STL) (Maler & Nickovic, 2004), using the following grammar:

$$\varphi := \text{true} \mid h(s) \geq 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2, \tag{1}$$

where $\varphi, \varphi_1, \varphi_2$ are STL formulae, and $h(s) \geq 0$ is a predicate over a signal $s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and a function $h : \mathbb{R}^n \to \mathbb{R}$. The Boolean operators $\neg$ and $\wedge$ denote negation and conjunction, respectively, and $\mathcal{U}_{[a,b]}$ denotes the bounded *until* operator, i.e. until between a and b, where $a < b$ and $a, b \in \mathbb{Q}_{\geq 0}$. Note that since $a, b \in \mathbb{Q}_{\geq 0}$, the STL formula inherently reasons over finite-time signals. We can also define other standard (temporal) operators from (1), such as disjunction $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, next $\bigcirc_a \varphi := \text{true}\, \mathcal{U}_{[a,a]} \varphi$, eventually $\Diamond_{[a,b]} \varphi := \text{true}\, \mathcal{U}_{[a,b]} \varphi$, and always $\Box_{[a,b]} \varphi := \neg\Diamond_{[a,b]} \neg\varphi$. The satisfaction relation $(s, t) \models \varphi$ indicates that the signal s starting at t satisfies φ. We consider the same definition of the semantics as in Roehm et al. (2016), which slightly deviates from e.g. Maler and Nickovic (2004) w.r.t. the *until* operator.[1] Since we build upon the results in Roehm et al. (2016), we have adopted the corresponding definition. STL is equipped with *quantitative semantics* $\rho(s, \varphi, t)$ that provides a robustness measure of how well a signal s starting at time t satisfies or violates the STL specification (Donzé & Maler, 2010; Fainekos & Pappas, 2009). If $\rho(s, \varphi, t)$ is negative, lower values imply that φ is more strongly violated. Conversely, if $\rho(s, \varphi, t)$ is positive, higher values imply that φ is satisfied more robustly.

2.2. Reachset temporal logic

Consider a closed-loop system described by:

$$\Sigma = \begin{cases} \dot{\xi}(t) = f_{\text{cl}}(t, \xi(t), \omega(t)), \\ \xi(0) \in I, \ \omega(t) \in \Omega, \end{cases} \tag{2}$$

where $\xi(t) \in \mathbb{R}^n$ denotes the state, $\omega(t) \in \Omega \subset \mathbb{R}^l$ an external disturbance, $I \subset \mathbb{R}^n$ is the set of initial conditions, I and Ω are compact, and $f : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$ and $\omega : \mathbb{R}_{\geq 0} \to \mathbb{R}^l$ are assumed to be Lipschitz continuous. In this work, we are not only interested in the STL performance of a single trajectory, but rather in the set of all trajectories satisfying system Σ, defined by

$$\mathcal{S}(\Sigma) := \{\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}^n \mid \forall t \geq 0 : \xi(t) \text{ satisfies } \Sigma\}. \tag{3}$$

That is, given an STL specification φ, we are interested in whether the system Σ satisfies $\forall \xi \in \mathcal{S}(\Sigma) : (\xi, t) \models \varphi$. For general systems of the form (2), it is not possible to construct the set $\mathcal{S}(\Sigma)$. However, it is possible to construct a set that does not persevere individual trajectories, but stores which states can be reached at a given time, i.e. a reachable set:

[1] In contrast to Maler and Nickovic (2004), our definition of the *until* operator does not require that φ_1 and φ_2 hold simultaneously.

Definition 1 (*Reachable Set*)**.** Given a system Σ, a mapping $R_e : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ is an *exact* reachable set if and only if:

$$\forall t \in \mathbb{R}_{\geq 0} : \{\xi(t) \mid \xi \in \mathcal{S}(\Sigma)\} = R_e(t). \tag{4}$$

A mapping $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ is a reachable set if and only if $\forall t \in \mathbb{R}_{\geq 0} : R_e(t) \subseteq R(t)$.

That is, a reachable set satisfies that $\forall t \in \mathbb{R}_{\geq 0}, \forall \xi \in \mathcal{S}(\Sigma) : \xi(t) \in R(t)$. Reachability analysis tools such as CORA (Althoff, 2015) can return a sequence of sets $\mathcal{R} = \mathcal{R}_{\{t_0\}}\mathcal{R}_{(t_0,t_1)}\mathcal{R}_{\{t_1\}}\mathcal{R}_{(t_1,t_2)} \ldots \mathcal{R}_{\{t_f\}}$, forming a reachable set given by

$$R(t) = \begin{cases} \mathcal{R}_{\{t_i\}} & \text{if } t = t_i, \\ \mathcal{R}_{(t_i,t_{i+1})} & \text{if } t \in (t_i, t_{i+1}). \end{cases} \tag{5}$$

As reachable sets do not store the information of individual trajectories, it is not possible to use the STL formula directly over the reachable sets. However, Roehm et al. (2016) introduced reachset temporal logic (RTL), enabling the direct reasoning over reachable sets, and established a sound transformation between STL and RTL. The RTL fragment relevant for this work is:

$$\psi := \text{true} \mid h(x) \geq 0 \mid \neg\psi \mid \psi_1 \wedge \psi_2,$$
$$\phi := \mathcal{A}\psi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \bigcirc_a \phi.$$

Here, ψ, ψ_1, ψ_2 are propositional formulae over states x and ϕ, ϕ_1, ϕ_2 are formulae over a reachable set $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$. Additionally, $\mathcal{A}$ denotes the *all* operator. The semantics of RTL is defined as follows:

$$\begin{aligned}
x &\models h(x) \geq 0 && \iff h(x) \geq 0, \\
x &\models \neg\psi && \iff x \not\models \psi, \\
x &\models \psi_1 \wedge \psi_2 && \iff x \models \psi_1 \text{ and } x \models \psi_2, \\
(R,t) &\models \mathcal{A}\psi && \iff \forall x \in R(t) : x \models \psi, \\
(R,t) &\models \phi_1 \vee \phi_2 && \iff (R,t) \models \phi_1 \text{ or } (R,t) \models \phi_2, \\
(R,t) &\models \phi_1 \wedge \phi_2 && \iff (R,t) \models \phi_1 \text{ and } (R,t) \models \phi_2, \\
(R,t) &\models \bigcirc_a \phi && \iff (R, t+a) \models \phi.
\end{aligned}$$

Consider the following notion:

Definition 2 (*c-divisible*)**.** An STL formula φ is said to be c-divisible, if all interval bounds of the temporal operators of φ are divisible by c.

Note that since $a, b \in \mathbb{Q}_{\geq 0}$, there always exists a c such that an STL formula is c-divisible. Given a c-divisible STL formula φ, the results in Roehm et al. (2016, Lemma 2 & Lemma 4) provide a sound transformation Υ_c to transform STL to RTL:

Theorem 1 (*Sound Transformation Roehm et al., 2016, Theorem 1*)**.** *Given the system Σ in (2), let φ be a c-divisible STL formula, and $R(t)$ be the reachable set of Σ in the form of (5) with $t_{i+1} - t_i = c$. The transformation Υ_c from Roehm et al. (2016), bringing the STL formula φ into an RTL formula $\phi = \Upsilon_c(\varphi)$, is sound, i.e.:*

$$\forall \xi \in \mathcal{S}(\Sigma) : (\xi, t) \models \varphi \iff (R, t) \models \phi. \tag{6}$$

The reachable set in (5) is formed by the reachable sequence $\mathcal{R}$, which partitions time into an alternating sequence of points and open intervals, whereas STL reasons over an infinite (but bounded) set of time instances. The transformation Υ_c first transforms the STL formula (rewritten in negation normal form) into sampled-time STL (Roehm et al., 2016): a subclass of STL that restricts to formulae with operators only reasoning over intervals $(0, c)$ and time shifts of fixed length c, such that the STL formula only reasons over an alternating sequence of points and open intervals. Here, the value $c/2$ can be seen as the time step between the points and a time interval. This transformation is sound, but not complete, i.e. for an STL formula φ and transformed sampled-time STL formula φ', we have $(\xi, t) \models \varphi \impliedby (\xi, t) \models \varphi'$, but the converse is not necessarily true. Subsequently, sampled-time STL (in conjunctive normal form) is transformed into RTL. In this transformation, the reasoning over trajectories is replaced with reasoning over a reachable set. The transformation between sampled-time STL to RTL is sound and complete. The transformation from STL to sampled-time STL results in general in an over-approximation, which can be reduced by taking smaller values of c. Since the full definition of the transformation Υ_c is quite involved, we refer the interested reader to Roehm et al. (2016).

The transformation Υ_c yields RTL formulae of the form

$$\phi = \bigwedge_{i \in \mathsf{I}} \bigvee_{j \in \mathsf{J}_i} \bigcirc_{j\frac{c}{2}} \bigvee_{k \in \mathsf{K}_{ij}} \mathcal{A}\psi_{ijk}, \tag{7}$$

where $\mathsf{I}, \mathsf{J}_i, \mathsf{K}_{ij}$ are finite index sets and ψ_{ijk} are non-temporal subformulae. As can be seen, j relates to a time step $c/2$, whereas i and k relate to the number of conjunctions and disjunctions. As an example, the transformation of $\varphi = \varphi_x \mathcal{U}_{[c,2c]} \varphi_v$ with $\varphi_x = x \geq 0$, $\varphi_v = v \geq 0$ is given by

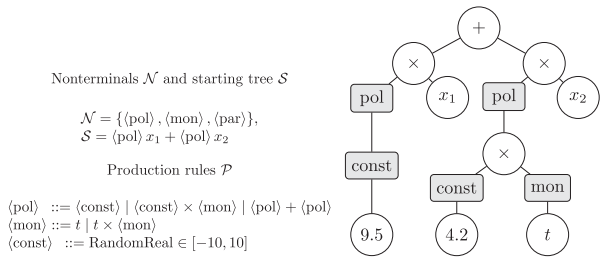
$$\begin{aligned}
\psi = {} & \bigcirc_0 \mathcal{A}\psi_x \wedge \bigcirc_{\frac{c}{2}} \mathcal{A}\psi_x \\
& \wedge \bigcirc_c \mathcal{A}(\psi_x \vee \psi_v) \wedge (\bigcirc_c \mathcal{A}\psi_v \vee \bigcirc_{\frac{3c}{2}} \mathcal{A}\psi_x) \\
& \wedge (\bigcirc_c \mathcal{A}\psi_v \vee \bigcirc_{\frac{3c}{2}} \mathcal{A}\psi_v \vee \bigcirc_{2c} \mathcal{A}(\psi_x \vee \psi_v)),
\end{aligned}$$

with $\psi_x = x \geq 0$, $\psi_v = v \geq 0$.

2.3. Genetic programming

The controllers in this work are synthesized using genetic programming (GP) (Koza, 1992), a variant of genetic algorithms (GA) (Holland, 1975), which evolves entire programs rather than optimizing parameters. In our case, the evolved program is a controller based on elementary building blocks consisting of state variables and basic functions, such as addition and multiplication. Within genetic programming, a candidate solution, called an *individual*, is represented by a data structure enabling easy manipulation, such as an expression tree. This data structure is called the *genotype*, whereas the individual itself, e.g., an analytic function, is referred to as the *phenotype*. A pool of individuals, called the *population*, is evolved based on a cost function, called the *fitness* function, which assigns a fitness score to all individuals. Depending on the fitness score, individuals can be selected to be recombined or modified using *genetic operators*, such as crossover and mutation. In the former case, two subtrees of individuals are interchanged, whereas in the latter case, a random subtree is replaced by a new random subtree. Each genetic operator has a user-defined rate, which determines the probability of the operator being applied to the selected individuals. A number of individuals are selected based on tournament selection: a fixed number of individuals are randomly selected from the population, and the individual with the highest fitness is returned. The process of selection and modification through genetic operators is repeated until a new population is created. The underlying hypothesis is that the average fitness of the population increases over many of these cycles, which are referred to as *generations*. The algorithm is terminated after a satisfying solution is found or a maximum number of generations is met.

We use the variant grammar-guided genetic programming (GGGP) (Verdier & Mazo, 2018; Whigham et al., 1995), which utilizes a grammar to which all individuals adhere: the population is initialized by creating random individuals adhering to the grammar and the used genetic operators are defined such that the resulting individuals also adhere to the grammar. The grammar is defined by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, where $\mathcal{N}$ is a set of nonterminals,



(a) Grammar (b) Genotype

Fig. 1. Example of a grammar and a genotype adhering to it. The corresponding phenotype is given by $9.5x_1 + 4.2tx_2$.

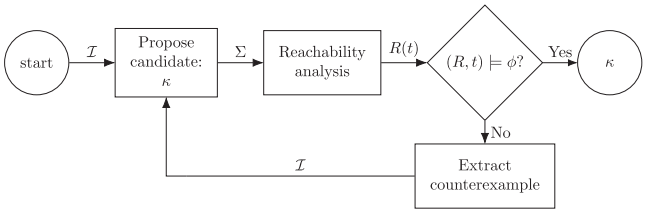


Fig. 2. Schematic overview of the algorithm.

$\mathcal{S}$ a starting tree, and $\mathcal{P}$ a set of production rules, which relate nonterminals to possible expressions. An example of a grammar is shown in Fig. 1a. In this grammar, the nonterminals correspond to polynomials $\langle \text{pol} \rangle$, monomials $\langle \text{mon} \rangle$ over time t, and constants $\langle \text{const} \rangle$. The starting tree $\mathcal{S}$ restricts the class of controllers to time-varying state feedback laws, linear in the state $x \in \mathbb{R}^2$. Given the grammar in Fig. 1a, an example of a genotype is shown in Fig. 1b, which has the corresponding phenotype of $9.5x_1 + 4.2tx_2$.

3. Problem definition and solution approach

We consider nonlinear systems subject to disturbances of the form:

$$\Sigma_{\text{ol}} = \begin{cases} \dot{\xi}(t) = f(t, \xi(t), u(t), \omega(t)), \\ \xi(0) \in I, \ \ \omega(t) \in \Omega, \end{cases} \tag{8}$$

with state $\xi(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, bounded disturbances $\omega(t) \in \Omega \subset \mathbb{R}^l$, and the set of initial conditions $I \subset \mathbb{R}^n$. The sets I and Ω are compact, and $f : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^n$, $u : \mathbb{R} \to \mathbb{R}^m$, and $\omega : \mathbb{R} \to \mathbb{R}^l$ are assumed to be Lipschitz continuous. We consider sampled-data time-varying state-feedback controllers $\kappa : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$ so that

$$u(t) = \kappa(t_k, \xi(t_k)) \text{ for all } t \in [t_k, t_k + \eta), \tag{9}$$

where t_k denotes the kth sampling instant, $t_0 = 0$, and η is the sampling time. This results in a closed-loop system of the form (2) for $t \in [t_k, t_k + \eta)$:

$$f_{\text{cl}}(t, \xi(t), \omega(t)) = f(t, \xi(t), \kappa(t_k, \xi(t_k)), \omega(t)). \tag{10}$$

The goal of this paper is formalized as follows:

Problem 3. Given a c-divisible STL formula φ, the open-loop system (8), and a sampling time η, synthesize a closed-form sampled-data time-varying controller $\kappa : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$ such that for all initial conditions and disturbances the resulting trajectories ξ of the closed-loop system satisfy φ, i.e.:

$$\forall \xi \in \mathcal{S}(\Sigma) : (\xi, 0) \models \varphi \tag{11}$$

In this work, we propose a counterexample-guided inductive synthesis (CEGIS) framework to synthesize a controller such that $(R, 0) \models \Upsilon_c(\varphi)$, thereby solving Problem 3 as follows from Theorem 1. The framework consists of iteratively proposing a controller obtained through GGGP[2] and then formally verifying the RTL formula $\Upsilon_c(\varphi)$ using reachability analysis. The proposed controller by GGGP is optimized w.r.t. a set of simulated trajectories (obtained through numerical integration), with the underlying idea that these are relatively fast to compute and provide a sensible search direction for the synthesis. The computationally more intensive reachability analysis verifies the resulting controller.

For a given open-loop system Σ_{ol}, c-divisible STL formula φ, reachability time step c, and grammar $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, the algorithm is initialized as follows:

(I1) The RTL formula ϕ is computed using $\phi = \Upsilon_c(\varphi)$ (see Theorem 1).
(I2) The set $\mathcal{I}$, consisting of pairs of initial conditions and disturbance realizations is initialized by randomly choosing n_s initial conditions $\{x^1, \ldots, x^{n_s}\} \subset I$, with random disturbance realizations $\omega^i : \mathbb{R}_{\geq 0} \to \Omega$, such that $\mathcal{I} = \left\{(x^1, \omega^1), \ldots, (x^{n_s}, \omega^{n_s})\right\}$.

Given the initial data, the algorithm goes through the following cycle, illustrated in Fig. 2, where each cycle is referred to as a *refinement*:

(A1) A candidate solution is proposed using GGGP, based on simulation trajectories corresponding to the set $\mathcal{I}$.
(A2) For the given candidate controller, the reachable set is computed.
(A3) Based on the reachable set, either:
  (a) $(R, t) \models \phi$, which is formally verified through SMT solvers, thus a controller solving Problem 3 is found.
  (b) $(R, t) \not\models \phi$, and a counterexample is extracted in the form of an initial condition x and a corresponding disturbance realization w. This pair (x, w) is added to $\mathcal{I}$ and the algorithm returns to step (A1).
  (c) $(R, t) \not\models \phi$ and a maximum of refinements is reached, therefore the algorithm is terminated.

To quantify the violation or satisfaction of an RTL formula, we introduce quantitative semantics for RTL in the next section. The proposal of a candidate controller in step (A1) is discussed in Section 5. The verification and counterexample generation in step (A3) is discussed in Section 6.

4. Quantitative semantics

Inspired by the quantitative semantics of STL (Donzé & Maler, 2010; Fainekos & Pappas, 2009), we define quantitative semantics for RTL in this section. These quantitative semantics provide a *robustness measure* on how well the formula is satisfied. For an RTL formula ϕ with propositional subformulae ψ, the quantitative semantics is given by functions $P(R, \phi, t)$ and $\varrho(x, \psi)$, respectively, recursively defined as:

$$\begin{aligned} \varrho(x, \text{true}) &= +\infty, \\ \varrho(x, h(x) \geq 0) &= h(x), \\ \varrho(x, \neg\psi) &= -\varrho(x, \psi), \\ \varrho(x, \psi_1 \wedge \psi_2) &= \min(\varrho(s, \psi_1), \varrho(s, \psi_2)), \\ P(R, \mathcal{A}\psi, t) &= \inf_{x \in R(t)} \varrho(x, \psi), \end{aligned}$$

[2] While GGGP evolves a population of controllers, only the controller with the highest fitness is returned.

$$P(R, \phi_1 \vee \phi_2, t) = \max(P(R, \phi_1, t), P(R, \phi_2, t)),$$
$$P(R, \phi_1 \wedge \phi_2, t) = \min(P(R, \phi_1, t), P(R, \phi_2, t)),$$
$$P(R, \bigcirc_a \phi, t) = P(R, \phi, t + a).$$

The quantitative semantics of STL are sound and complete (Donzé, Ferrère, & Maler, 2013; Fainekos & Pappas, 2009). The quantitative semantics of RTL also have these properties:

Theorem 2 (*Soundness and Completeness*). *Let ϕ be an RTL formula, R a reachable set, and t a time instance, then:*

(1) $P(R, \phi, t) > 0 \Rightarrow (R, t) \models \phi$ *and* $(R, t) \models \phi \Rightarrow P(R, \phi, t) \geq 0$,
(2) $P(R, \phi, t) < 0 \Rightarrow (R, t) \not\models \phi$ *and* $(R, t) \not\models \phi \Rightarrow P(R, \phi, t) \leq 0$.

Remark 4. Note that $P(R, \phi, t) = 0$ does not imply $(R, t) \models \phi$ nor $(R, t) \not\models \phi$. This is because on the boundary of an inequality, the distinction between inclusion or exclusion is lost within the quantitative semantics. That is, if $\varrho(x, \psi) = 0$, we also have $\varrho(x, \neg\psi) = 0$, hence the quantitative semantics of two mutually exclusive logic formulae evaluate to the same value.

The proof of Theorem 2 can be found in the Appendix. Consider a c-divisible STL formula φ and the corresponding RTL formula $\phi = \Upsilon_c(\varphi)$ in the form of (7). Using the equivalences $\bigcirc_a(\phi_1 \wedge \phi_2) = \bigcirc_a\phi_1 \wedge \bigcirc_a\phi_2$ and rewriting ψ_{ijk} in disjunctive normal form, we can express the RTL formula (7) as:

$$\phi = \bigwedge_{i \in \mathsf{I}} \bigvee_{j \in \mathsf{J}_i, k \in \mathsf{K}_{ij}} \phi'_{ijk}, \tag{12a}$$

$$\phi'_{ijk} = \bigcirc_{j\frac{c}{2}} \mathcal{A} \bigvee_{a \in \mathsf{A}^{ijk}} \bigwedge_{b \in \mathsf{B}_a^{ijk}} h_{ab}^{ijk}(x) \sim 0, \tag{12b}$$

where A^{ijk} and B_a^{ijk} denote finite index sets, $\sim \in \{\geq, >\}$, and $h_{ab}^{ijk}(x) \sim 0$ is a predicate over x. Using the quantitative semantics, the robustness measure of this RTL formula for a reachable set R and time $t = 0$ is given by

$$P(R, \phi, 0) = \min_{i \in \mathsf{I}} \left(\max_{j \in \mathsf{J}_i, k \in \mathsf{K}_{ij}} P(R, \phi'_{ijk}, 0) \right), \tag{13a}$$

$$P(R, \phi'_{ijk}, 0) = \inf_{x \in R(j\frac{c}{2})} \left(\max_{a \in \mathsf{A}^{ijk}} \left(\min_{b \in \mathsf{B}_a^{ijk}} h_{ab}^{ijk}(x) \right) \right). \tag{13b}$$

5. Candidate controller synthesis

In this section, we detail step (A1) of the proposed algorithm in Section 3, i.e., the proposal of a candidate controller. The candidate controller is synthesized using GGGP, by maximizing an approximation of the robustness measure, based on a finite number of simulated trajectories. The sampling time is equal to $c/2$ to coincide with the time instances at which the robustness measure $P(R, \phi, 0)$ is evaluated. For an RTL formula of the form (7) and $t = 0$, the first and the final time instances of relevance τ_0 and τ_f, are given by $\tau_0 = 0$ and $\tau_\mathrm{f} = \frac{c}{2} \max_{i \in \mathsf{I}} |\mathsf{J}_i|$, respectively. Let us denote the finite set of sampled-time instances $\hat{T} = \{\tau_0, \ldots, \tau_\mathrm{f}\}$. Given a candidate controller $\kappa : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$, a set of pairs of initial conditions and disturbance realizations $\mathcal{I}$, we consider an approximated reachable set $\hat{R}^\kappa_\mathcal{I} : \hat{T} \to 2^{\mathbb{R}^n}$ formed by all corresponding simulated trajectories $x : \hat{T} \to \mathbb{R}^n$, such that for a given time instance $\tau_q \in \hat{T}$:

$$\hat{R}^\kappa_\mathcal{I}(\tau_q) = \{x(\tau_q) \mid (x(\tau_0), \omega) \in \mathcal{I}\}.$$

Provided $\hat{R}^\kappa_\mathcal{I}$, we approximate the robustness measure by $P(\hat{R}^\kappa_\mathcal{I}, \phi, 0)$.

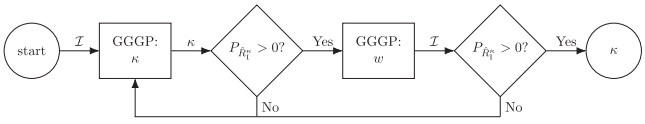


Fig. 3. Schematic overview of the synthesis of candidate controller.

5.1. Outline of the candidate controller synthesis

The proposal of a candidate controller in step (A1) is based on approximating an optimal controller that solves:

$$\sup_\kappa \inf_\mathcal{I} \; P(\hat{R}^\kappa_\mathcal{I}, \phi, 0). \tag{14}$$

If the optimum is positive, i.e. the optimal controller yields a positive robustness measure w.r.t. the worst-case approximated reachability set, it can be expected that $P(R, \phi, 0)$ is positive,[3] which would imply from Theorems 1 and 2 that the corresponding optimal controller solves Problem 3. To (approximately) solve this optimization problem, our algorithm alternatively updates the controller and the disturbances within $\mathcal{I}$, as described in the following steps, which are also illustrated in Fig. 3:

(A1.a) Given the set $\mathcal{I}$, We synthesize an analytic expression $\kappa : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ by using GGGP to solve:

$$\arg\sup_\kappa P(\hat{R}^\kappa_\mathcal{I}, \phi, 0). \tag{15}$$

If for the resulting controller κ the robustness measure approximation $P(\hat{R}^\kappa_\mathcal{I}, \phi, 0)$ is negative, this optimization step in (15) is repeated. Otherwise, the algorithm continues to the next step.

(A1.b) Given the controller κ, for each initial condition x^i in $\mathcal{I}$, an analytic expression for a disturbance realization $\omega^i : \mathbb{R} \to \Omega$ is synthesized using GGGP, in which the robustness measure approximation is minimized, i.e.:

$$\omega^* = \arg\sup_{\omega^i} \quad -P\left(\hat{R}^\kappa_{\{(x^i, \omega^i)\}}, \phi, 0\right). \tag{16}$$

The set $\mathcal{I}$ is then updated by replacing (x^i, ω^i) with (x^i, ω^*), i.e.

$$\mathcal{I} \leftarrow (\mathcal{I} \backslash \{(x^i, \omega^i)\}) \cup \{(x^i, \omega^*)\}.$$

If the corresponding robustness degree approximation $P(\hat{R}^\kappa_{\{(x^i, \omega^i)\}}, \phi, 0)$ is negative, the algorithm returns to step (A1.a). Otherwise, if for all updated disturbance realizations the robustness measure approximation is positive, i.e., $\forall i,\ P(\hat{R}^\kappa_{\{(x^i, \omega^i)\}}, \phi, 0) > 0$, the algorithm returns a candidate controller.

5.2. Reference-tracking controllers

To speed up the synthesis, it is possible to impose a structure to the solution. In this section we discuss the design of reference-tracking controllers, based on a nominal reference trajectory $x_\mathrm{ref}(t)$ and a corresponding feedforward input $u_\mathrm{ff}(t)$. That is, we consider a time-varying reference-tracking controller of the form:

$$\kappa(t, x) = u_\mathrm{ff}(t) + \kappa_\mathrm{fb}(t, x - x_\mathrm{ref}(t)), \tag{17}$$

[3] Due to e.g. truncation errors in the numerical integration, there can be a mismatch between $\inf_\mathcal{I} P(\hat{R}^\kappa_\mathcal{I}, \phi, 0)$ and $P(R, \phi, 0)$. Therefore, the proposed framework always relies on additional formal verification.

where $\kappa_{\text{fb}} : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbb{R}^m$ is a time-varying feedback controller. This controller is then used in a sampled-data fashion as in (9). The feedforward input and reference trajectory can be computed beforehand as follows:

(R1) Given a point $x_0 \in int(I)$, (e.g. the centroid of I if I is convex), an analytic expression for $u_{\text{ff}} : \mathbb{R} \to \mathbb{R}^m$ is synthesized using GGGP, by maximizing the approximated robustness measure for a nominal trajectory starting at x_0, i.e. a trajectory with no disturbance:

$$\arg\sup_{u_{\text{ff}}} \quad P(\hat{R}^{u_{\text{ff}}}_{\{(x_0, \mathbf{0}_l)\}}, \phi, 0).$$

(R2) Given the feedforward input u_{ff}, an analytic expression for the corresponding nominal reference trajectory $x_{\text{ref}} : \mathbb{R} \to \mathbb{R}^n$ is synthesized using GGGP. Given the simulated solution $x(\tau_k)$ corresponding to $x(0) = x_0$, $\omega(t) = \mathbf{0}_l$, and $u(t) = u_{\text{ff}}(t)$, x_{ref} is obtained by fitting for each state dimension $i \in \{1, \dots, n\}$ an expression to $x_i(\tau_k)$, based on the Euclidean norm of the error vector $e_i = [e_i(\tau_0), \dots, e_i(\tau_f)]$, with $e_i(\tau_k) = x_i(\tau_k) - x_{\text{ref},i}(\tau_k)$, i.e., maximizing:

$$\arg\sup_{x_{\text{ref},i}} (1 - \|e_i\|)^{-1}.$$

Using the synthesized pair $(u_{\text{ff}}(t), x_{\text{ref}}(t))$, the user-defined grammar within GGGP can be used to enforce the structure of a time-varying reference controller in (17) within step (A1), as demonstrated by the following brief example:

Example 5. Let us consider a one-dimensional system with dimensions $n = m = 1$. The structure of (17), where we further restrict κ to be linear in state, can be enforced by taking the starting tree $\mathcal{S} = u_{\text{ff}}(t) + \langle \text{pol} \rangle (x - x_{\text{ref}})$ and the production rules from Fig. 1a.

6. Reachability analysis and verification

In this section we detail step (A3) of the algorithm. We use polynomial zonotopes $\mathcal{PZ}$ as the set representation of the reachable set[4]:

Definition 6 (*Polynomial Zonotope*)**.** Given a generator matrix $G \in \mathbb{R}^{n \times h}$ and exponent matrix $E \in \mathbb{Z}^{p \times h}_{\geq 0}$, a polynomial zonotope $\mathcal{PZ}$ is defined as

$$\mathcal{PZ} := \left\{ \sum_{i=1}^{h} \left(\Pi_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1, 1] \right\},$$

where $E_{(k,i)}$ denotes the ith entry the kth row of E and $G_{(\cdot,i)}$ denotes the ith column of G. The vector $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_p]^T$ is referred to as the *parameterization vector* of the polynomial zonotope.

Consider a parameterization vector $\boldsymbol{\alpha}$ and a reachable set $R(t)$ expressed as a polynomial zonotope. The corresponding point in the reachable set $z(\boldsymbol{\alpha}, R(t)) \in \mathbb{R}^n$ is given by:

$$z(\boldsymbol{\alpha}, R(t)) = \sum_{i=1}^{h} \left(\Pi_{k=1}^{p} \alpha_k^{E^R_{(k,i)}} \right) G^R_{(\cdot,i)}, \tag{18}$$

where E^R and G^R denote the exponent matrix and generator matrix of $R(t)$, respectively. The benefit of polynomial zonotopes as set representation is that dependencies between points in subsequent reachable sets are maintained under the reachability analysis operations (Kochdumper et al., 2020). That is, for a reachable set $R : \mathbb{R}_{\geq 0} \to 2^{\mathbb{R}^n}$ and parameterization vector $\boldsymbol{\alpha}$, we have $\xi(t) = z(\boldsymbol{\alpha}, R(t)) \implies \xi(0) = z(\boldsymbol{\alpha}, R(0))$. This enables the extraction of an initial condition corresponding to a point for which the specification is violated. Using this method, we construct a counterexample in the form of a pair of initial condition and disturbance realization (x_0, ω), such that the corresponding trajectory results in a violation of the RTL formula. After reachability analysis, the algorithm undergoes the following steps:

(B1) For all subformulae ϕ'_{ijk} in (12b), the corresponding robustness sub-score (13b) is computed by solving the following nonlinear optimization problem[5] over the corresponding set $R(jc/2)$:

$$p^*_{ijk} = \inf_{\boldsymbol{\alpha}_{ijk}} \left(\max_{a \in \mathsf{A}^{ijk}} \left(\min_{b \in \mathsf{B}^{ijk}_a} h^{ijk}_{ab}(z(\boldsymbol{\alpha}_{ijk}, R\left(\tfrac{jc}{2}\right))) \right) \right). \tag{19}$$

(B2) Given the robustness sub-scores p^*_{ijk}, compute the full robustness measure (13a):

$$p^* = \min_{i \in \mathsf{I}} \max_{j \in \mathsf{J}_i, k \in \mathsf{K}_{ij}} p^*_{ijk}. \tag{20}$$

(B3) As we rely on nonlinear optimization, we cannot guarantee to find the global optimum p^*. However, as it is a minimization problem, a suboptimal solution is an upperbound $\hat{p}$, such that $P(R, \phi, 0) = p^* \leq \hat{p}$. Given $\hat{p}$, either:

(a) $\hat{p} < 0$, hence the RTL specification is violated. In this case, given the argument $(ijk)^*$ solving (20), we extract an initial condition x_0 corresponding to $\boldsymbol{\alpha}_{(ijk)^*}$, i.e. $x_0 = z(\boldsymbol{\alpha}_{(ijk)^*}, R(0))$. For this initial condition x_0, a disturbance realization ω is synthesized similarly to step (A1.b), i.e., GGGP is used to solve:

$$\arg\sup_{\omega} -P\left(\hat{R}^{\kappa}_{\{(x_0,\omega)\}}, \phi, 0\right).$$

The counterexample given by the pair (x_0, ω) is subsequently added to $\mathcal{I}$. This new set $\mathcal{I}$ is then used to improve upon the synthesized controller in step (A1).

(b) $\hat{p} \geq 0$, hence the RTL specification is potentially satisfied. However, to guarantee this, we perform an additional verification step, based on Satisfiability Modulo Theories (SMT) solvers (Barrett, Sebastiani, Seshia, & Tinelli, 2009), which are capable of verifying first-order logic formulae. The subformula (12b) holds if the following first-order logic formula holds:

$$\forall x \in R\left(j\frac{c}{2}\right) : \bigvee_{a \in \mathsf{A}^{ijk}} \bigwedge_{b \in \mathsf{B}^{ijk}_a} h^{ijk}_{ab}(x) \sim 0, \tag{21}$$

where again $\sim \in \{\geq, >\}$. Suitable SMT solvers to verify (21) include Z3 (de Moura & Bjørner, 2008) when $R(jc/2)$ and h^{ijk}_{ab} are expressed as polynomials, and dReal (Gao, Kong, & Clarke, 2013) when these are expressed as general nonlinear expressions.[6] Given the Boolean answers to the subformulae in (12b) for all i, j, and k, it is trivial to compute the Boolean answer to (12a).

[4] In this definition, without loss of generality and for the ease of exposition, we only consider (dependent) generators G and we omit independent generators; for the full definition we refer to Kochdumper and Althoff (2020).

[5] To use gradient-based optimization, max and min can be approximated by $M^{\beta}_{a \in \mathsf{A}}(x_a) = (\sum_{a \in \mathsf{A}} x_a e^{\beta x_a})/(\sum_{a \in \mathsf{A}} e^{\beta x_a})$, where A denotes an iterator set and for $\beta \to \infty$, $M^{\beta}_{a \in \mathsf{A}}(x_a) \to \max_{a \in \mathsf{A}} x_a$ and $\beta \to -\infty$, $M^{\beta}_{a \in \mathsf{A}}(x_a) \to \min_{a \in \mathsf{A}} x_a$.

[6] dReal implements a δ-complete decision procedure (Gao, Avigad, & Clarke, 2012). If the reachable set is robust w.r.t. the RTL formula, this has no consequence for our proposed framework.

Table 1
General settings for each of the case studies. The number of individuals, GGGP generations, and CMA-ES generations are shown for each controller component and disturbance realizations.

System	n_s	Individuals				GGGP generations				CMA-ES generations			
		u_{ff}	x_{ref}	κ	ω^i	u_{ff}	x_{ref}	κ	ω^i	u_{ff}	x_{ref}	κ	ω^i
Car	7	14	14	14	14	30	10	3	3	20	10	10	3
Path planning	10	28	28	14	14	30	50	3	3	40	40	10	3
Aircraft	5	28	42	14	14	50	50	5	5	40	60	10	3
Platoon	10	–	–	14	14	–	–	3	3	–	–	10	3
Spacecraft	7	–	–	14	14	–	–	5	5	–	–	10	3

A synthesized controller, formally verified in step (B3.b), solves Problem 3, as formalized in the following theorem:

Theorem 3 (*Correct-by-design Controller*)**.** *Given a c-divisible STL formula φ, an open-loop system (8), and a sampling time η, if the algorithm in Section 3 returns a controller before the maximum number of refinements, then the closed-loop system satisfies*

$$\forall \xi \in \mathcal{S}(\Sigma) : (\xi, 0) \models \varphi. \tag{22}$$

Proof. If the algorithm terminates, the returned controller results in a reachable set R such that $(R, 0) \models \psi$, where $\phi = \Upsilon_c(\varphi)$. By Theorem 1, we have $\forall \xi \in \mathcal{S}(\Sigma) : (\xi, 0) \models \varphi$. □

7. Dealing with conservatism

Conservatism, in the reachability analysis and the transformation from STL to RTL, may cause that $(R, 0) \not\models \phi$, whereas $\forall \xi(0) \in I$, $(\xi, 0) \models \varphi$, i.e., the desired STL specification holds for all initial conditions, whereas based on the reachability set, the RTL specification is not met. This conservatism can be reduced by refining settings such as the time steps or Taylor order in the reachability tool (see Althoff, 2013), or reducing the parameter c to obtain less conservative RTL formulae ϕ, at the cost of increased overall computational complexity. Similarly, truncation errors of the integration scheme, and conservatism within reachability analysis (introducing spurious trajectories) can lead to mismatches between $\hat{R}^{\kappa}_{\mathcal{I}}$ and $R(t)$. This mismatch can be bridged considering an optional error signal ε added to the simulated trajectory $x(\tau_q)$, which is co-synthesized with the disturbance realizations, as will be shown in the subsequent case studies.

Issues due to conservatism can also be dealt with within the synthesis of a candidate controller in step (A1), e.g. the controllers within GGGP could be further optimized w.r.t. the robustness, such that the added robustness could potentially compensate for conservative reachability analysis. Controller complexity (measured as the number of nonterminals) can also be used as a secondary optimization criterion to facilitate less conservative reachability analysis. The resulting multi-objective optimization problem is solved using Pareto-optimality ranking (Deb, Pratap, Agarwal, & Meyarivan, 2002) that results in a rank that is used as the new fitness value used within the selection.

Finally, note that the optimization problems in steps (A1.a), (A1.b), (R1), (R2), (B1), and (B3.a) are non-convex and therefore finding a global optimum cannot be guaranteed. The optimization problems are used to propose candidate controllers or to provide counterexamples that constrain the solution space. As the goal is to find a qualitatively correct controller rather than an (quantitatively) optimal one, loss in optimality is of a lesser importance. Moreover, by the use of mutation within genetic programming, the algorithm is capable of exploring the search space, until a solution is found.

Table 2
Production rules $\mathcal{P}$.

$\mathcal{N}$	Rules
$\langle \text{expr}_t \rangle$	$::= \langle \text{pol}_t \rangle \mid \langle \text{pol}_t \rangle \times \langle \text{trig}_t \rangle \mid \langle \text{expr}_t \rangle + \langle \text{expr}_t \rangle$
$\langle \text{trig}_t \rangle$	$::= \tanh(\langle \text{pol}_t \rangle) \mid \sin(\langle \text{pol}_t \rangle) \mid \cos(\langle \text{pol}_t \rangle)$
$\langle \text{pol}_t \rangle$	$::= 0 \mid \langle \text{const} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon}_t \rangle \mid \langle \text{pol}_t \rangle + \langle \text{pol}_t \rangle$
$\langle \text{mon}_t \rangle$	$::= t \mid t \times \langle \text{mon}_t \rangle$
$\langle \text{pol}_x \rangle$	$::= \langle \text{const} \rangle \times \langle \text{mon}_x \rangle \mid \langle \text{pol}_x \rangle + \langle \text{pol}_x \rangle$
$\langle \text{mon}_x \rangle$	$::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{mon}_x \rangle$
$\langle \text{var} \rangle$	$::= x_1 \mid \cdots \mid x_n$
$\langle \text{const} \rangle$	$::=$ Random Real $\in [-1, 1]$

8. Case studies

In this section we demonstrate the effectiveness of the proposed framework on benchmarks from competing synthesis methods, i.e. reachability-based (Schürmann & Althoff, 2017a) (car example), MPC-based (Lindemann & Dimarogonas, 2017) (path planning), and abstraction-based (Reissig et al., 2017) (airplane landing maneuver). Moreover, we consider the effect of input saturation, which is enforced through the STL specification. Additionally, we consider a platooning benchmark (Schürmann & Althoff, 2017b) to investigate the scalability w.r.t. system dimension. While we use for the aforementioned benchmarks the reference-tracking controller structure discussed in Section 5.2, we demonstrate the ability to synthesize controllers from scratch on a simplified spacecraft (Brockett, 1983) in Section 8.6.

The case studies are performed using an Intel Xeon CPU E5-1660 v3 3.00 GHz using 14 parallel CPU cores. The GGGP algorithm is implemented in Mathematica 12 and the reachability is performed using CORA in MATLAB. Motivated by the non-convex and discontinuous nature of the optimization problems, we use population-based optimization methods, but any suitable optimization tool can be used instead. For the optimization problem in (19), we use particle swarm optimization of the global optimization toolbox in MATLAB. Within each generation of GGGP, parameters within an individual are optimized using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen & Ostermeier, 2001), based on the same fitness function as used for GGGP. More specifically, we use the variant sep-CMA-ES (Ros & Hansen, 2008), due to its linear space and time complexity. For the verification of (21), we use the SMT solver dReal with $\delta = 0.001$.

Across all benchmarks, the probability rate of the crossover and mutation operators being applied on a selected individual are 0.2 and 0.8, respectively. Benchmark-specific settings are shown in Table 1, which include the number of simulations n_s, number of individuals, and the number of GGGP and CMA-ES generations. Note that the number of GGGP generations for κ and ω^i is the number of generations per step (A1.a) and (A1.b), and not the total of GGGP generations per proposal of a controller in step (A1), which depends on the number of times step (A1.a) and (A1.b) are repeated. For each case study, we use a grammar with nonterminals and production rules as shown in Table 2. These

nonterminals correspond to general time-dependent expressions $\langle \text{expr}_t \rangle$, time-dependent trigonometric functions $\langle \text{trig}_t \rangle$, time- and state-dependent polynomial expression $\langle \text{pol}_t \rangle$ and $\langle \text{pol}_x \rangle$, respectively, time- and state-dependent monomials $\langle \text{mon}_t \rangle$ and $\langle \text{mon}_x \rangle$, respectively, variables $\langle \text{var} \rangle$, and constants $\langle \text{const} \rangle$. The polynomials are restricted to polynomials over either time t or states x, where the state-dependent polynomials are further restricted to not contain zero degree monomials. The time-dependent expressions are formed by time-dependent polynomials, a product of these polynomials, and time-dependent trigonometric functions, and a sum of two expressions. The trigonometric functions are restricted to hyperbolic tangents, sines and cosines with time-dependent polynomial arguments. Note that per case study, different starting trees are used, such that potentially only a subset of the grammar is available. E.g., if the starting tree is $\langle \text{pol}_t \rangle$, candidate solutions are restricted to time-dependent polynomial solutions.

We use Runge–Kutta as numerical integration scheme. To keep a constant number of initial conditions in $\mathcal{I}$, counterexamples are added using a first-in, first-out principle. To compensate for the gap between the simulation and the reachability analysis (as discussed in Section 7), we consider an added error signal bounded by the scaled vector field of the dynamics f, parameterized by

$$\varepsilon(t, x) = \delta\sigma(t) f(t, x(t), u(t), \omega(t)), \tag{23}$$

where δ is a constant and $\sigma : \mathbb{R}_{\geq 0} \to [-1, 1]^{n \times n}$ a time-varying diagonal matrix which determines the sign and magnitude of the error signal. The constant δ is optimized after each reachability analysis such that the mismatch between the robustness measure and the approximated robustness measure is minimized, i.e.:

$$\arg\inf_{\delta} \left\| P(R, \phi, 0) - P\left(\hat{R}^{\kappa}_{\{(x,\omega)\}}, \phi, 0\right) \right\|, \tag{24}$$

where $\{(x, \omega)\}$ is the counterexample pair computed in Section 6.

In reporting the synthesized controllers, its parameters are rounded from six to three significant numbers for space considerations.

Finally, in this section we denote the logic function indicating set membership of a set Y by φ_Y, i.e. given a set Y in the form

$$Y := \left\{ x \in \mathbb{R}^n \;\middle|\; \bigvee_i \bigwedge_j h_{ij}(x) \sim 0 \right\}, \quad \sim \in \{\geq, >\},$$

where $h_{ij} : \mathbb{R}^n \to \mathbb{R}$, we have $\varphi_Y = \bigvee_i \bigwedge_j h_{ij}(x) \sim 0$.

8.1. Car benchmark

Let us consider a kinematic model of a car from (Schürmann & Althoff, 2017a):

$$\begin{cases} f(x, u, \omega) = (u_1 + w_1, u_2 + w_1, x_1 \cos(x_2), x_1 \sin(x_2))^T, \\ I = [19.9, 20.2] \times [-0.02, 0.02] \times [-0.2, 0.2]^2, \\ \Omega = [-0.5, 0.5] \times [-0.02, 0.02]. \end{cases}$$

where the states x_1, x_2, x_3, x_4 denote the velocity, orientation, and x and y position of the car, respectively. Furthermore, u_1 and u_2 denote the inputs and w_1 and w_2 disturbances. The sampling time η of the sampled-data controller is set to be 0.025 s. Similarly to Schürmann and Althoff (2017a), we consider a "turn left" maneuver over a time interval $T = [0, 1]$, where within T, the trajectories stay within the safe set S and at the final time instant, the system is in the goal set, captured by the STL specification:

$$\varphi_1 = \square_{[0,1]} \varphi_S \wedge \square_{\{1\}} \varphi_G. \tag{25}$$

We consider the following safe set S and goal set G:

$$S = [19.5, 20.5] \times [-0.1, 0.3] \times [-1, 25] \times [-1, 5],$$

$$G = [19.95, 20.05] \times [0.18, 0.22] \times [19.85, 19.9] \times [1.98, 2].$$

To guide the synthesis, we impose the reference-tracking controller structure from Section 5.2 and therefore we first design a feedforward signal and reference trajectory using GGGP. For u_{ff}, x_{ref}, we use polynomial expressions as a function of time t, for the feedback law κ we restrict the search space to reference-tracking controllers which are linear in the tracking error and polynomial in time:

$$\kappa(x, t) = u_{\text{ff}}(t) + K(t)(x - x_{\text{ref}}(t)), \tag{26}$$

and for ω^i we consider saturated polynomials in time. This is done using the grammar with starting trees:

$$\mathcal{S}_{u_{\text{ff}}} = (\langle \text{pol}_t \rangle, \langle \text{pol}_t \rangle)^T, \quad \mathcal{S}_{x_{\text{ref},i}} = \langle \text{pol}_t \rangle,$$

$$\mathcal{S}_{\kappa} = u_{\text{ff}} + \begin{pmatrix} \langle \text{pol}_t \rangle, \ldots, \langle \text{pol}_t \rangle \\ \langle \text{pol}_t \rangle, \ldots, \langle \text{pol}_t \rangle \end{pmatrix} (x - x_{\text{ref}}),$$

$$\mathcal{S}_{\omega^i} = \left(\text{sat}_{(\underline{\omega}_1, \overline{\omega}_1)}(\langle \text{pol}_t \rangle), \text{sat}_{(\underline{\omega}_2, \overline{\omega}_2)}(\langle \text{pol}_t \rangle) \right)^T.$$

Here, $\text{sat}_{(\underline{\omega}_i, \overline{\omega}_i)}$ denotes a saturation function such that $\omega^i(t) \in \Omega$, where $\text{sat}_{(\underline{\omega}_i, \overline{\omega}_i)}(x) = \max(\underline{\omega}_i, \min(x, \overline{\omega}_i))$. Finally, for each disturbance realization, we co-evolve the error signal ε^i in (23), which is dependent on the candidate controller κ and disturbance realization ω^i:

$$\mathcal{S}_{\varepsilon^i} = \delta\sigma f(t, x, \kappa(x), \omega^i),$$

$$\sigma = \text{diag}(\text{sat}_{(-1,1)}(\langle \text{pol}_t \rangle), \ldots, \text{sat}_{(-1,1)}(\langle \text{pol}_t \rangle)),$$

where diag denotes a diagonal matrix. For the simulations and reachability analysis, we use a sampling time of 0.025 s and 0.0125 s, respectively.

First, a feedforward control input and reference trajectory for a nominal initial condition are synthesized as described in Section 5.2. An example of a found feedforward controller and corresponding reference trajectory are shown in Table 4. For 10 independent runs, the average synthesis time of u_{ff} and the reference trajectory per dimension $x_{\text{ref},i}$ is shown in Table 3. Using these u_{ff} and x_{ref} as building blocks for the controller, κ is synthesized as described in step (A1). An example of a synthesized $K(t)$ in (26) is given by

$$K(t) = \begin{pmatrix} -41.5 & -6.48t^2 & -84.3958 & 9.45 \\ 3.58 & -30.1 & -8.22 & 3.62t1 - 49.2t^2 \end{pmatrix}.$$

The corresponding reachable set is shown in Fig. 4. We observe that the final reachable set is not within the goal set. The red dots represent the violation and the corresponding initial condition. After refining the controller iteratively, an example of a controller satisfying φ_1 after 3 refinements is shown in Table 4.

For 10 independent synthesis runs of κ, statistics on the number of generations, number of refinements, complexity in terms of number of non-terminals, and computation time is shown in Table 3 and Fig. 7. In most cases, a solution was obtained around 3 refinements. However, due to the stochastic nature of the approach, in one case it took 20 refinements before a solution was found.

8.2. Input saturation

In our general framework, we do not canonically consider input saturation. Input saturation can be considered in multiple ways, such as restricting the grammar of the controller to include a saturation function, or even a continuous approximation using e.g. a sigmoid function. However, the downside of such an approach is that the reachability analysis under these functions is typically challenging for state-of-the-art reachability tools, due to the strong nonlinearity or hybrid nature. Instead, for illustrative

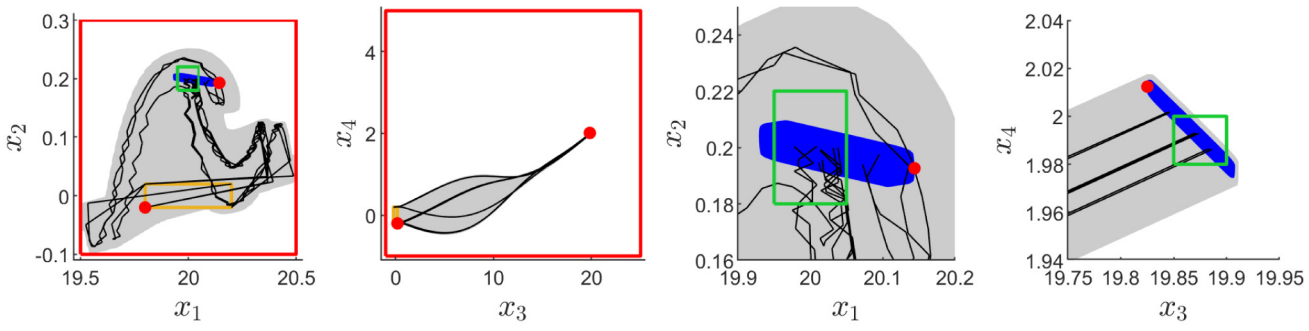


Fig. 4. Reachable set for the first controller for the car benchmark, which violates the desired controller specification. Figures (c) and (d) illustrate the reachable set near the goal set. Red dots: a point in the final reachable set that is outside of the goal set and its corresponding initial state, yellow: initial set, green: goal set G, gray: reachable set, red: safe set S, blue: reachable set at $t = 1$, black: example of simulation traces. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

purposes, we incorporate the constraint within the STL specification, such that for all states in the reachable set the saturation bounds are not exceeded. Let us revisit the car benchmark, where we consider the same input constraints as in Schürmann and Althoff (2017a), namely $u \in \overline{U} = [-9.81, 9.81] \times [-0.4, 0.4]$. The STL specification is extended to:

$$\varphi_2 = \varphi_1 \wedge \square_{[0,1]} \varphi_U \tag{27}$$

with

$$U = \left\{ x \in \mathbb{R}^n \mid \kappa(x) \in \overline{U} \right\}. \tag{28}$$

The synthesis statistics are shown in Table 3 and Fig. 7. An example of a synthesized $K(t)$ in (26) is given by

$$K(t) = \begin{pmatrix} -18.1 + 18.2t - 65.9t^6 & 0.22t \\ 0 & -8.26 - 41.8t \\ -29.6 - 48.7t & 0 \\ -11.2t & -33.1t^2 \end{pmatrix}^T.$$

In most cases, a solution was found in around 4 to 5 refinements, with the exceptions of two runs with 20 and 40 refinements, respectively.

8.3. Path planning for simple robot

Let us consider the path-planning problem for a simple robot adopted from Lindemann and Dimarogonas (2017). We deviate from Lindemann and Dimarogonas (2017) in considering the system in continuous time and consider bounded disturbances. The system is described by:

$$\begin{cases} f(x, u, \omega) = (u_1 + w_1, u_2 + w_2, x_1, x_2)^T, \\ I = \{0\}^2 \times [0.5, 1.5]^2, \ \ \Omega = [-0.05, 0.05]^2, \end{cases}$$

where the state vector represents the x-velocity, y-velocity, x-position and y-position, respectively. The sampling time of the sampled-data controller η is set to be 0.5 s. Similar to Lindemann and Dimarogonas (2017), we consider the specification in which the system needs to remain in a safe set S and eventually visit regions P_1, P_2 and P_3:

$$\varphi' = \square_{[0,25]} \phi_S \wedge \Diamond_{[5,25]} \phi_{P_1} \wedge \Diamond_{[5,25]} \phi_{P_2} \wedge \Diamond_{[5,25]} \phi_{P_3}, \tag{29}$$

with $S = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [0, 10]^2\}$, $P_1 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [8, 10]^2\}$, $P_2 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [8, 10] \times [0, 2]\}$, $P_3 = \{x \in \mathbb{R}^n \mid (x_3, x_4) \in [0, 2] \times [8, 10]\}$. In Lindemann and Dimarogonas (2017), the input is constrained s.t. $u \in \overline{U} = [-1, 1]^2$. Similar to Section 8.2, we impose this constraint through the STL specification, yielding the following STL specification:

$$\varphi = \varphi' \wedge \square_{[0,25]} \varphi_U, \tag{30}$$

where U is given by (28). We consider the same controller structure and grammar as the previous benchmark, with the exception of the grammar of the feedforward input and reference trajectory. For these elements, we extend the grammar to expressions which can include trigonometric functions, by using the grammar in Table 2 and the starting trees $\mathcal{S}_{u_{\text{ff}}} = (\langle \text{expr}_t \rangle, \langle \text{expr}_t \rangle)$ and $\mathcal{S}_{x_{\text{ref},i}} = \langle \text{expr}_t \rangle$. For the simulations and reachability analysis, we use a sampling time of 0.5 s. The statistics on the synthesis is again shown in Table 3 and Fig. 7. An example of the controller elements u_{ff}, x_{ref} and $K(t)$ of a synthesized controller are shown in Table 4. The corresponding reachable set of the state and input is shown in Fig. 5. Across 10 independent runs, commonly in 1 to 2 refinements a solution was found, with one run requiring 8 refinement.

8.4. Landing maneuver

Let us consider the landing aircraft maneuver, adopted from Reissig et al. (2017). The system model is given by

$$\begin{cases} f(x, \nu, \omega) = \begin{pmatrix} \frac{1}{m}(\nu_1 \cos \nu_2 - D(\nu_2, x_1) - mg \sin x_2) \\ \frac{1}{mx_1}(\nu_1 \sin \nu_2 + L(\nu_2, x_1) - mg \cos x_2) \\ x_1 \sin x_2 \end{pmatrix}, \\ D(\nu_2, x_1) = (2.7 + 3.08(1.15 + 4.2\nu_2)^2) x_1^2, \\ L(\nu_2, x_1) = (68.6(1.25 + 4.2\nu_2)) x_1^2, \\ \nu_i = u_i + w_i, \quad i = 1, 2, \\ I = [80, 82] \times [-2°, -1°] \times \{55\} \\ \Omega = [-5 \cdot 10^3, -5 \cdot 10^3] \times [-0.25°, 0.25°], \end{cases}$$

where the states x_1, x_2, x_3 denote the velocity, flight path angle and the altitude of the aircraft, ν_i denotes a disturbed input, where u_1 denotes the thrust of the engines and u_2 the angle of attack. Finally, $D(\nu, x_1)$ and $L(\nu, x_1)$ denote the lift and drag, respectively, and $m = 60 \cdot 10^3$ kg, $g = 9.81$ m/s^2. The sampling time of the sampled-data controller is set at $\eta = 0.25$ seconds. Compared to Reissig et al. (2017), we do not consider measurement errors, but the proposed framework can be adapted arbitrarily to accommodate this type of disturbance. We define the following safe set, goal set and input bounds:

$$\begin{aligned} S =& [58, 83] \times [-3°, 0°] \times [0, 56], \\ G =& [63, 75] \times ([-2°, -1°] \times [0, 2.5]) \\ & \cap \{x \in \mathbb{R}^3 \mid x_1 \sin x_2 \geq -0.91\}, \\ \overline{U} =& [0, 160 \cdot 10^3] \times [0°, 10°], \end{aligned}$$

and consider the following specification:

$$\varphi = (\varphi_S \wedge \varphi_U) \mathcal{U}_{[18,20]} \varphi_G, \tag{31}$$

Table 3
Statistics over an average of 10 independent synthesis runs. Time FF: average computation time of the feedforward components; Total gen.: total number of GGGP generations for κ before a solution was found; Total ref.: total number of refinements; Complexity: number of total non-terminals within the genotype of the synthesized controller; GP κ: synthesis of candidate κ using GGGP; GP ω: disturbance realization optimization; RA: reachability analysis; CE: counterexample extraction; SMT: verifying the specification through an SMT solver; min: minimum; med: median; max: maximum. The average contribution percentages do not sum up to one, as the contribution of routines such as writing (SMT) files is not displayed.

System	Time FF [s]		Total gen.			Total ref.			Complexity			Time [min]			Average contribution to total time [%]				
	u_{ff}	$x_{ref,i}$	min	med	max	min	med	max	min	med	max	min	med	max	GP κ	GP ω	RA	CE	SMT
Car	45.1	1.2	63	205.5	1410	3	6	19	14	27	69	16.5	41.6	204.1	37.9	26.2	3.15	19.3	3.44
Constrained car	–	–	84	318	933	2	5	8	24	35.5	56	28.0	61.2	117.0	42.5	17.2	1.70	15.8	9.19
Path planning	254.0	19.1	3	16.5	117	1	2.5	9	8	11.5	15	14.1	23.8	61.8	7.61	9.50	3.05	17.2	27.8
Aircraft	708.2	46.2	45	342.5	1165	2	5	16	24	36	58	44.0	165.1	422.8	36.7	22.5	12.9	10.3	7.71
Platoon N = 2	–	–	4	64.5	171	1	2	7	12	21.5	42	3.44	9.30	30.7	29.0	33.0	3.42	20.5	1.48
Platoon N = 3	–	–	522	1611	3210	4	5	8	30	34	56	67.6	207.9	398.6	60.7	32.4	0.851	3.12	0.403
Spacecraft	–	–	5	67.5	1350	1	2.5	24	14	14	22	3.79	22.35	378.3	25.0	37.3	2.15	18.13	1.28

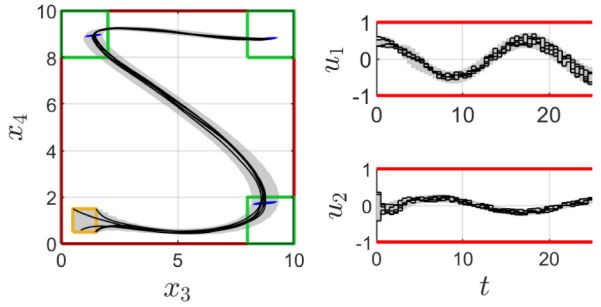


Fig. 5. Reachable set of a found controller for the path planning benchmark. (a) Reachable set of the x–y position. (b) Reachable set of the input over time. Yellow: initial set, gray: reachable set, red: safe set S and input constraints, green: target sets P_1, P_2, and P_3, black: selection of simulated trajectories, blue: reachable sets at certain time instances within one of the target sets. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

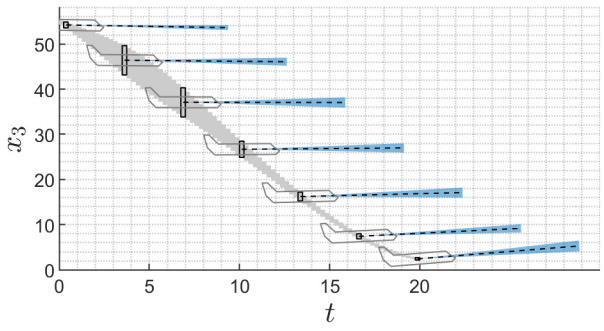


Fig. 6. Time evolution of the reachable set of the altitude x_3 under a synthesized controller for the landing maneuver. Gray: Reachable set over time of the altitude x_3. Blue: the set of the aircraft pitch $x_2 + u_2$ for 7 time intervals. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where the set U is given by (28). That is, trajectories are always within the safe set and satisfy the input constraints, until between 18 and 20 s the goal set is reached.

We use the same controller structure and grammar as the path-planning problem. For the simulations and reachability analysis, we use a sampling time of 0.25 s. The algorithm settings are shown in Table 1. The statistics of 10 independent synthesis runs are again shown in Table 3 and Fig. 7. An example of the controller elements u_{ff}, x_{ref} and $K(t)$ of a synthesized controller is shown in Table 4. The corresponding reachable set of the altitude over time, as well as the reachable sets of the pitch angles at multiple time instances are shown in Fig. 6.

8.5. Scalability: platoon

Consider a platooning system (Schürmann & Althoff, 2017b), described by:

$$\begin{cases} f(x,u,w) = (f_1(x,u,w),\dots,f_N(x,u,w))^T, \\ f_i(x,u,w) = (x_{2i},\, \delta u_i + \delta w_i),\ i = \{1,\dots,N\}, \\ \delta u_1 = u_1,\ \delta w_1 = w_1, \\ \delta u_i = u_{i-1} - u_i,\ w_i = w_{i-1} - w_i,\ i = \{2,\dots N\}, \\ I = [-0.2, 0.2] \times [19.8, 20.8] \\ \qquad \times ([0.8, 1.2] \times [-0.2, 0.2])^{N-1}, \\ \Omega = [-1, 1]^N, \end{cases}$$

where N denotes the number of vehicles, x_1, x_2 the position and velocity of the first vehicle, x_{2i}, x_{2i+1} the relative position and relative velocities of vehicle i, and the input u_i denotes the acceleration of vehicle i. We consider a sampling time η of 0.05 s. The specification involves the acceleration of the platoon up to a goal set within a second, subjected to input constraints, while each vehicle maintains a safe distance, which is captured by the STL formula

$$\varphi = \square_{[0,1]} \left(\varphi_s \wedge \varphi_{\overline{U}}\right) \wedge \square_{\{1\}} \varphi_G, \tag{32}$$

where $\varphi_s = \bigwedge_{i=2}^{N} x_{2i-1} \geq 0$, $G = [20.8, 21.2] \times [21.5, 22.5] \times ([0.8, 1.2] \times [-0.2, 0.2])^{N-1}$, and $\overline{U} = [-10, 10]^N$. We use the feed forward signal $u_{ff,i} = 1.4$ for all $i \in \{1,\dots,N\}$, where we exploit the fact that the desired control input for the agents after the first one is equal to the first one. The reference trajectory for this feedforward controller is given by:

$$x_{ref}(t) = (20.3t + 0.7t^2, 20.3 + 1.4t, 1, 0, \dots, 1, 0)^T.$$

We impose the structure $\kappa_i(x,t) = \kappa_{i-1}(x,t) - \kappa_i'(x,t)$ for $i = 2,\dots,N$, where for $\kappa_i(x,t)$ and $\kappa_i'(x,t)$ we use the same control the same controller structure and grammar as the path-planning problem. For the simulations and reachability analysis, we use a sampling time of 0.05 s. The algorithm settings are shown in Table 1.

Given a maximum of 5000 GGGP generations, for $N = 2$, in 10 independent runs a controller was found, for $N = 3$, in 9 out of 10 runs, and for $N = 4$, no solutions were found. The results statistics for the successful synthesis runs are again shown in Table 3 and Fig. 7.

8.6. Discovering structures from scratch: spacecraft

Let us consider a simplified model of spacecraft (Brockett, 1983), described by:

$$\begin{cases} f(x,u,w) = \begin{pmatrix} u_1 + w_1, & u_2 + w_2 & x_1 x_2 \end{pmatrix}^T, \\ I = [-0.5, 0.5]^2 \times [1, 2],\ \Omega = [-0.1, 0, 1]^2, \end{cases}$$

where states denote the angular velocity and the inputs denote control torques that are aligned with the principle axes. We consider a sampling time η of 0.1 s. For the simulations and reachability analysis, we use a sampling time of 0.1 s. Note that

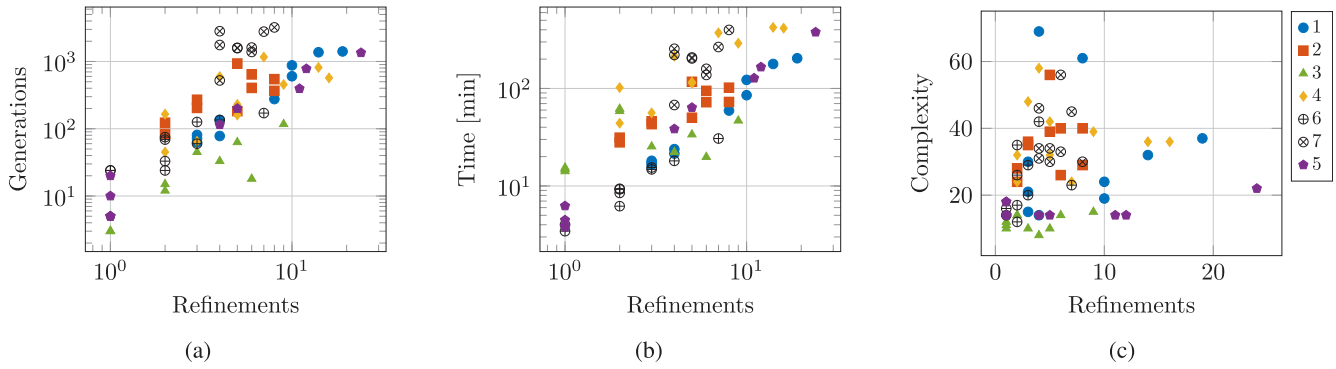


Fig. 7. Number of refinements versus (a) number of GGGP generations, (b) time in minutes, and (c) complexity of the controller, measured in number of non-terminals, for systems (1) car, (2) constrained car, (3) path planning, (4) aircraft, (5) platoon $N = 2$, (6) platoon $N = 3$, (7) spacecraft.

Table 4
Examples of synthesized controllers. Numerical values are rounded for space considerations.

System	Car (without input constraints)	Path planning	Aircraft
u_{ff}	$\begin{pmatrix} 0.01835 \\ 0.1995 \end{pmatrix}$	$\begin{pmatrix} 0.500\cos(0.362t + 0.0733) \\ -0.190\sin(0.678 - 0.324t) \end{pmatrix}$	$\begin{pmatrix} 255.68 + 107.57t^2 \\ 0.00956 + 0.00419t \end{pmatrix}$
x_{ref}	$\begin{pmatrix} 19.999 + 0.020567t \\ 0.19954t \\ 19.981t - 0.10838t^4 \\ 1.9915t^2. \end{pmatrix}$	$\begin{pmatrix} 0.03t - 3.81\cos(0.361t) + 4.72 \\ 1.38\sin(0.361t) + 0.024 \\ 0.406t + 1.88\cos(0.312t + 0.949), \\ 0.427 - 0.583\cos(0.765 - 0.325t) \end{pmatrix}$	$\begin{pmatrix} 81.5 - 0.380t - 1.28\sin(0.393 + 0.164t) \\ (-0.164 - 1.59 \cdot 10^{-3}t)\cos(0.103t) + 0.138\cos(0.120t) \\ 55.7 - 0.674\cos(0.354t) - 2.96t\sin(0.788 + 0.062t) \end{pmatrix}$
$K(t)$	$\begin{pmatrix} -43.4 & 3.94 & -89.6 & 307.3t^2 \\ -8.28t^5 & -33.3 & -6.21 & -10.1 \end{pmatrix}$	$\begin{pmatrix} -0.264 & -0.125t & 0 & 0.209t \\ 0 & -0.204 & -0.781 & -1.35 \end{pmatrix}$	$\begin{pmatrix} -2.67t^3 & -0.407 - 0.0636t & -0.788 - 0.461t \\ -0.00607 & -0.0217 - 0.237t - 0.0348t^2 & -0.00023t^2 \end{pmatrix}$

for stabilization, linearization methods are not appropriate, as the system linearized around points in the set $\{x \in \mathbb{R}^3 \mid x_1, x_2 = 0\}$ are not controllable. The goal is to control the system in finite time to a set around the origin $G = [-0.2, 0.2]^3$ and the control input is constrained s.t. $u \in U = [-5, 5]^2$, which is captured by the STL specification

$$\varphi = \square_{[0,5]}\phi_U \wedge \square_{\{5\}}\varphi_G.$$

For the disturbance, we use the same grammar as in the previous case studies. For the controller we consider polynomial state feedback controllers This is done using the starting tree $\mathcal{S}_u = (\langle \text{pol}_x \rangle, \langle \text{pol}_x \rangle)$. The algorithm settings are shown in Table 1. The results of 10 independent synthesis runs are again shown in Table 3 and Fig. 7. An example of a found controller is given by

$$\kappa(x) = (-2.056x_1 - 2.233x_3, -2.034x_2 + 2.071x_3)^T.$$

Of the 10 synthesized controllers, 8 controllers have the same structure as the above controller.

9. Discussion

We discuss now the results from Section 8 and relate them to the results in the literature. A candidate solution is proposed using Recall that a GGGP *generation* is the cycle of creating a new population through fitness evaluation, selection and applying genetic operators. A *refinement* is defined as the cycle of proposing a candidate solution based on GGGP, validation using reachability analysis, and extracting counterexamples. Therefore, in each refinement, there are one or multiple GGGP generations. First of all, Fig. 7a shows a polynomial relation between the number of refinements and the total number of GGGP generations. Secondly, Fig. 7b shows a polynomial relation between the number of refinements versus the total computation time. Finally, Fig. 7c illustrates that more refinements do not imply that complexity of the controller increases. However, the complexity of the found controller does seem to be dependent on the system and STL specification.

While the computation time is related to the number of refinements, this relationship depends on the STL specification and the dynamics. For the car benchmark without and with input constraints, we observe that the added constraints within the STL specification increased the required number of generations, and typically required more time per refinement. Hence, the total computation time heavily depends on the STL specification, as expected. Additionally, we observe an increase in the median of the complexity of the resulting controllers.

With the platoon example, we see that for an increase in state dimension the number of generations required to find a solution significantly increases. This is expected, as the search space is significantly larger. For $N = 4$, no solutions were found within 5000 GGGP generations. However, it is worth nothing that the optimal solution for $N = 4$ in Schürmann and Althoff (2017b) very tightly satisfies specification. However, general conclusions regarding computation time and system order cannot be drawn. For example, the input-constrained car and path planning benchmarks are both four-dimensional systems, where the STL specification of the latter is more involved. Regardless, the path-planning problem has a lower computation time and requires less generations and number of refinements, indicating a dependency between the computation time and the dynamics of the system, which is also as expected.

The resulting offline time complexity of our proposal is clearly higher than alternative methods like the one in Schürmann and Althoff (2017a), with synthesis time for the car benchmark of just around 10 s, or in Reissig et al. (2017), taking about 700 s for the aircraft benchmark controller synthesis. Both these alternative methods are faster offline, at the cost of potentially much larger controllers to be stored: a linear controller for each sampling time in Schürmann and Althoff (2017a); an exponentially growing number of entries in a look-up table with increasing system dimension in discretization-based methods like that of Reissig et al. (2017). However, for the systems for which synthesis is successful in both ours and discretization-based methods, the size of the resulting controllers does not seem prohibitive. Nonetheless, a more clear advantage of our approach is the ability to constrain

the controller structure so as to produce controllers that are easier to *understand* by end-users than e.g. the look-up-tables (or BDDs) of discretization-based methods.

An alternative to alleviate the memory footprint of controllers is the use of MPC approaches, such as Lindemann and Dimarogonas (2017) for the path-planning problem. These solutions require additional online computational complexity compared to the quick evaluation that our controllers enable. As an example, the most complex of our aircraft controllers just requires on average $4.7 \cdot 10^{-7}$ s to compute the control actions (evaluated using `timeit` in MATLAB running on an Intel i7-8750H CPU).

In summary, the tests performed indicate that this approach may be competitive with respect to competing alternatives whenever the application at hand requires both low memory and computational footprint, but more importantly whenever there is a need to impose specific controller structures to improve interpretability of the controller.

As most automated synthesis methods for the type of problems we handle, the proposed framework is not a complete method. That is, the method is not guaranteed to find a solution in a finite number of iterations, regardless of its existence. Nevertheless, for the presented case studies, in 10 independent runs a solution was always found. Since the search space is navigated nondeterministically, we observed that the number of GGGP generations, number of refinements and computation time can vary significantly for each run.

As it has been highlighted, the offline computation time of our current implementation is not competitive with those in other references. Note however, that the performance of our implementation has not been optimized for speed, being a mix of Matlab and Mathematica code, or parallelization of individuals in GGGP, which consume most of the computation, c.f. Table 3. Additionally, limiting the fragment of STL, e.g., to $h(s)$ linear, the robustness degree computation can be considerably simplified. If the robustness measure is also upper bounded in a non-conservative manner, the usage of SMT solvers becomes redundant. This would significantly reduce the computation time for benchmarks such as the path-planning problem. Finally, input constraints, currently part of the STL formula, can also be captured using saturation functions in our grammar, simplifying the synthesis. However, as a caveat, discontinuous functions such as saturation functions significantly complicate the reachability analysis.

10. Conclusion

We have proposed a framework for CEGIS-based correct-by-design controller synthesis for STL specifications based on reachability analysis and GGGP. The effectiveness has been demonstrated based on a selection of case studies. While the synthesis time is outmatched by methods solving similar problems, the proposed method results in a compact closed-form analytic controller which is provably correct when implemented in a sampled-data fashion. This enables the implementation in embedded hardware with limited memory and computation resources.

Acknowledgment

The authors would like to thank Bastian Schürmann for fruitful discussions on this work.

Appendix. Proof of Theorem 2

Theorem 2 is proven by induction over the structure of the RTL formula ϕ and subformula ψ. This is only done for the first statement in Theorem 2, as the second statement is logically equivalent to the first, i.e.:

$$P(R,\phi,t) > 0 \Rightarrow (R,t) \models \phi \equiv (R,t) \not\models \phi \Rightarrow P(R,\phi,t) \leq 0,$$
$$(R,t) \models \phi \Rightarrow P(R,\phi,t) \geq 0 \equiv P(R,\phi,t) < 0 \Rightarrow (R,t) \not\models \phi.$$

Moreover, the proof for the cases $\psi = \text{true}$, $\psi = h(x) \geq 0$, $\psi = \neg\psi_1$, $\psi = \psi_1 \wedge \psi_2$, $\phi = \phi_1 \wedge \phi_2$, and $\phi = \phi_1 \vee \phi_2$ is analogous to the proof of Proposition 16 in Fainekos and Pappas (2009), in which similar conditions are proven for Metric Temporal Logic, and are therefore omitted. The full proof can be found in the extended paper in Verdier, Kochdumper, Althoff, and Mazo (2020).

- **Case** $\phi = \mathcal{A}\psi$: For this formula ϕ, the quantitative semantics is given by $P(R,\mathcal{A}\psi,t) = \inf_{x\in R(t)} \varrho(x,\psi)$. **(i)** If $P(R,\mathcal{A}\psi,t) > 0$, then $\forall x \in R(t) : \varrho(x,\psi) > 0$. By the induction hypothesis, $\forall x \in R(t) : x \models \psi$, thus from the semantics we have $(R,t) \models \mathcal{A}\psi$. **(ii)** If $(R,t) \models \mathcal{A}\psi$, then from the semantics we have $\forall x \in R(t) : x \models \psi$. By the induction hypothesis, we get $\forall x \in R(t) : \varrho(x,\psi) \geq 0$, thus $P(R,\mathcal{A}\psi,t) \geq 0$.
- **Case** $\phi = \bigcirc_a \phi_1$: For this formula ϕ, the quantitative semantics is given by $P(R,\bigcirc_a\phi_1,t) = P(R,\phi,t+a)$. **(i)** If $P(R,\bigcirc_a\phi_1,t) > 0$, then $P(R,\phi_1,t+a) > 0$. By the induction hypothesis, we get $(R,t+a) \models \phi_1$, thus from the semantics we have $(R,t) \models \bigcirc_a\phi_1$. **(ii)** If $(R,t) \models \bigcirc_a\phi_1$, then from the semantics we have $(R,t+a) \models \phi_1$. By the induction hypothesis, we get $P(R,\bigcirc_a\phi_1,t) \geq 0$. □

References

Abate, A., Bessa, I., Cattaruzza, D., Cordeiro, L., David, C., Kesseli, P., et al. (2017). Automated formal synthesis of digital controllers for state-space physical plants. In *Computer aided verification* (pp. 462–482). Springer.

Aksaray, D., Jones, A., Kong, Z., Schwager, M., & Belta, C. (2016). Q-learning for robust satisfaction of signal temporal logic specifications. In *IEEE conf. on decision and control (CDC)* (pp. 6565–6570).

Althoff, M. (2010). *Reachability analysis and its application to the safety assessment of autonomous cars* (Ph.D. dissertation), Technische Universität München.

Althoff, M. (2013). Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Hybrid systems: Computation and control* (pp. 173–182).

Althoff, M. (2015). An introduction to CORA 2015. In *Proc. of the workshop on applied verification for continuous and hybrid systems*.

Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. MIT Press.

Barrett, C. W., Sebastiani, R., Seshia, S. A., & Tinelli, C. (2009). Satisfiability modulo theories. In *Handbook of satisfiability, Vol. 185* (pp. 825–885). IOS Press.

Belta, C., & Sadraddini, S. (2019). Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), 115–140.

Belta, C., Yordanov, B., & Gol, E. A. (2017). *Formal methods for discrete-time dynamical systems, Vol. 89*. Springer.

Brockett, R. W. (1983). Asymptotic stability and feedback stabilization. *Differential Geometric Control Theory*, *27*(1), 181–191.

de Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. In *Tools and algorithms for the construction and analysis of systems* (pp. 337–340). Springer Berlin Heidelberg.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Ding, J., Li, E., Huang, H., & Tomlin, C. J. (2011). Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *IEEE int. conf. on robotics and automation* (pp. 2160–2165).

Donzé, A., Ferrère, T., & Maler, O. (2013). Efficient robust monitoring for STL. In *Computer aided verification* (pp. 264–279). Springer Berlin Heidelberg.

Donzé, A., & Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. In *Formal modeling and analysis of timed systems* (pp. 92–106). Springer Berlin Heidelberg.

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

Farahani, S. S., Raman, V., & Murray, R. M. (2015). Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine*, *48*(27), 323–328, Analysis and Design of Hybrid Systems (ADHS).

Gao, S., Avigad, J., & Clarke, E. M. (2012). δ-Complete decision procedures for satisfiability over the reals. In *Int. joint conf. on automated reasoning* (pp. 286–300). Springer.
Gao, S., Kong, S., & Clarke, E. M. (2013). DReal: An SMT solver for nonlinear theories over the reals. In *Int. conf. on automated deduction* (pp. 208–214). Springer.
Garg, K., & Panagou, D. (2019). Control-Lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications. In *IEEE conf. on decision and control (CDC)* (pp. 1422–1429).
Girard, A., Pola, G., & Tabuada, P. (2010). Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, *55*(1), 116–126.
Habets, L. C. G. J. M., Collins, P. J., & van Schuppen, J. H. (2006). Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, *51*(6), 938–948.
Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.
Holland, J. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.
Kochdumper, N., & Althoff, M. (2020). Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 1.
Kochdumper, N., Schürmann, B., & Althoff, M. (2020). Utilizing dependencies to obtain subsets of reachable sets. In *Hybrid systems: Computation and control*. ACM.
Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. MIT Press.
Lindemann, L., & Dimarogonas, D. V. (2017). Robust motion planning employing signal temporal logic. In *2017 American control conf. (ACC)* (pp. 2950–2955).
Lindemann, L., & Dimarogonas, D. V. (2019a). Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, *3*(1), 96–101.
Lindemann, L., & Dimarogonas, D. V. (2019b). Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks. *Automatica*, *106*, 284–293.
Lindemann, L., Verginis, C. K., & Dimarogonas, D. V. (2017). Prescribed performance control for signal temporal logic specifications. In *IEEE conf. on decision and control (CDC)* (pp. 2997–3002).
Liu, W., Mehdipour, N., & Belta, C. (2021). Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*, 1.
Liu, J., Ozay, N., Topcu, U., & Murray, R. M. (2013). Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control*, *58*(7), 1771–1785.
Maler, O., & Nickovic, D. (2004). Monitoring temporal properties of continuous signals. In *Formal techniques, modelling and analysis of timed and fault-tolerant systems* (pp. 152–166). Springer.
Pant, Y. V., Abbas, H., Quaye, R. A., & Mangharam, R. (2018). Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In *ACM/IEEE int. conf. on cyber-physical systems (ICCPS)* (pp. 186–197).
Raman, V., Donzé, A., Sadigh, D., Murray, R. M., & Seshia, S. A. (2015). Reactive synthesis from signal temporal logic specifications. In *Hybrid systems: Computation and control* (pp. 239–248). ACM.
Ravanbakhsh, H., & Sankaranarayanan, S. (2015). Counter-example guided synthesis of control Lyapunov functions for switched systems. In *IEEE conf. on decision and control (CDC)* (pp. 4232–4239).
Reissig, G., Weber, A., & Rungger, M. (2017). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, *62*(4), 1781–1796.
Roehm, H., Oehlerking, J., Heinz, T., & Althoff, M. (2016). STL model checking of continuous and hybrid systems. In *Automated technology for verification and analysis* (pp. 412–427). Springer Int. Publishing.
Ros, R., & Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel problem solving from nature – PPSN X: 10th int. conf., Dortmund, Germany, September 13-17, 2008. Proc.* (pp. 296–305). Springer Berlin Heidelberg.
Sadraddini, S., & Belta, C. (2015). Robust temporal logic model predictive control. In *2015 53rd annual Allerton conf. on communication, control, and computing* (pp. 772–779).
Sadraddini, S., & Belta, C. (2018). Formal guarantees in data-driven model identification and control synthesis. In *Hybrid systems: Computation and control* (pp. 147–156). ACM.
Schürmann, B., & Althoff, M. (2017a). Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. In *Proc. of the 20th IFAC world congress* (pp. 12020–12027).
Schürmann, B., & Althoff, M. (2017b). Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *2017 American control conference (ACC)* (pp. 2522–2529). IEEE.
Schürmann, B., Kochdumper, N., & Althoff, M. (2018). Reachset model predictive control for disturbed nonlinear systems. In *IEEE Conf. on Decision and Control (CDC)* (pp. 3463–3470).
Schürmann, B., Vignali, R., Prandini, M., & Althoff, M. (2020). Set-based control for disturbed piecewise affine systems with state and actuation constraints. *Nonlinear Analysis. Hybrid Systems*, *36*, Article 100826.
Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., & Saraswat, V. (2006). Combinatorial sketching for finite programs. In *Proc. of the 12th int. conf. on architectural support for programming languages and operating systems* (pp. 404–415). ACM.
Tabuada, P. (2009). *Verification and control of hybrid systems: A symbolic approach*. Springer US.
Verdier, C. F., Kochdumper, N., Althoff, M., & Mazo, M., Jr. (2020). Formal synthesis of closed-form sampled-data controllers for nonlinear continuous-time systems under STL specifications. arXiv preprint arXiv:2006.04260.
Verdier, C. F., & Mazo, M., Jr. (2018). Formal synthesis of analytic controllers for sampled-data systems via genetic programming. In *IEEE conf. on decision and control (CDC)* (pp. 4896–4901).
Verdier, C. F., & Mazo, M., Jr. (2020). Formal controller synthesis for hybrid systems using genetic programming. arXiv preprint arXiv:2003.14322.
Whigham, P. A., et al. (1995). Grammatically-based genetic programming. In *Proc. of the workshop on genetic programming: from theory to real-world applications* (pp. 33–41).
Yaghoubi, S., & Fainekos, G. (2019). Worst-case satisfaction of STL specifications using feedforward neural network controllers: A Lagrange multipliers approach. *ACM Transactions on Embedded Computing Systems*, *18*(5s).
Zapreev, I. S., Verdier, C., & Mazo, M., Jr. (2018). Optimal symbolic controllers determinization for BDD storage. *IFAC-PapersOnLine*, *51*(16), 1–6.

Cees Ferdinand Verdier is a control and dynamics engineer at Hardt Hyperloop. He received his Ph.D. and M.Sc. degrees in Systems and Control in 2020 and 2015 respectively, and a B.Sc. degree in Mechanical Engineering in 2013, all from the Delft University of Technology, the Netherlands. From 2020 to 2021 he was a postdoctoral researcher at the same university. His main interests are hybrid systems, formal methods for control, and computational intelligence.

Niklas Kochdumper received the B.S. degree in Mechanical Engineering in 2015 and the M.S. degree in Robotics, Cognition and Intelligence in 2017, both from the Technical University of Munich, Germany. He is currently pursuing the Ph.D. degree in computer science at the Technical University of Munich, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, computational geometry, controller synthesis, and electrical circuits.

Matthias Althoff is an associate professor in Computer Science at the Technical University of Munich, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from the Technical University of Munich, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include the formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, robotics, and power systems.

Manuel Mazo Jr. is an associate professor at the Delft Center for Systems and Control, Delft University of Technology (The Netherlands). He received the Ph.D. and M.Sc. degrees in Electrical Engineering from the University of California, Los Angeles, in 2010 and 2007 respectively. He also holds a Telecommunications Engineering "Ingeniero" degree from the Polytechnic University of Madrid (Spain), and a "Civilingenjör" degree in Electrical Engineering from the Royal Institute of Technology (Sweden), both awarded in 2003. Between 2010 and 2012 he held a joint post-doctoral position at the University of Groningen and the innovation center INCAS3 (The Netherlands). His main research interest is the formal study of problems emerging in modern control system implementations, and in particular the study of networked control systems and the application of formal verification and synthesis techniques to control.