Formal synthesis of analytic controllers for sampled-data systems via genetic programming

Verdier, Cees F.; Mazo, Manuel

DOI
10.1109/CDC.2018.8619121

Publication date
2018

Document Version
Final published version

Published in
Proceedings of the 57th IEEE Conference on Decision and Control (CDC 2018)

2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Formal Synthesis of Analytic Controllers for Sampled-Data Systems via Genetic Programming

Cees F. Verdier and Manuel Mazo Jr

***Abstract*— This paper presents an automatic formal controller synthesis method for nonlinear sampled-data systems with safety and reachability specifications. Fundamentally, the presented method is not restricted to polynomial systems and controllers. We consider a periodically switched controllers based on a Control Lyapunov Barrier-like function. The proposed method utilizes genetic programming to synthesize these function in analytic form, as well as the controller modes. Correctness of the controller are subsequently verified by means of a Satisfiability Modulo Theories solver. Effectiveness of the proposed methodology is demonstrated on multiple systems.**

I. INTRODUCTION

Modern controller design for nonlinear continuous systems often involves both reachability and safety specifications. Furthermore, digital controller implementations typically impose that states are measured periodically and that control signals are held constant in between sampling. This paper proposes an approach to automatically synthesize periodically switched state feedback controllers for a special subclass of safety and reachability specifications for nonlinear sampled-data systems.

Two popular paradigms for automatic controller synthesis for reachability and safety specifications are: 1) abstraction and simulation, and 2) Control Lyapunov functions (CLF) and Control Barrier Functions (CBF).

The first approach abstracts the infinite system to a finite one, which simplifies the formal controller synthesis for temporal logic specifications [1]. For nonlinear systems, tools implementing this approach include PESSOA [2], SCOTS [3] and CoSyMa [4]. The second approach deals with the system as an infinite system. Control Lyapunov functions [5] and Control Barrier Functions [6] are design tools for stabilization and safety specifications respectively. In [7] and [8] attempts are made to combine both CLFs and CBFs. Automatic synthesis of these functions is often done by posing the problem as a sum of squares (SOS) problem, which can be solved through convex optimization, see e.g. [9] and [10]. Drawbacks of the abstraction and (bi-)simulation approach are that it requires discretization of the state space and that the resulting controller is often an enormous look-up table in the form of a sparse matrix or a binary decision diagram (BDD). On the other hand, the SOS programming paradigm is limited to polynomial systems. Although reformulation of some nonpolynomial systems to an SOS formulation exists, e.g. [9], [11] and references therein, polynomial Lyapunov functions can be too restrictive, as global asymptotical stability of a polynomial system does not imply the existence of a polynomial Lyapunov function [12].

To overcome these limitations, we propose a framework which uses genetic programming (GP) in combination with a Satisfiability Modulo Theories (SMT) solver. GP is an evolutionary algorithm which evolves encoded representations of symbolic functions [13], rather than just fitting optimal parameters given a predefined structure. An SMT solver is a tool which uses a combination of background theories to determine whether a first-order logic formula can be satisfied [14]. Our approach uses a CLF-like function and a predefined switching law that infers a reachability and safety specification. The proposed framework uses GP to automatically generate both candidate CLFs and optionally the controller modes of a periodically switched state feedback controller. The SMT solver is subsequently used to formally verify the candidate solutions. By using GP, we allow ourselves to search for solutions that include nonpolynomial functions. Furthermore, the synthesized controllers are expressed as analytic expressions that are significantly more compact than BDDs returned by abstraction-based methods.

This work is a follow-up to [15], in which also a combination of GP and SMT solvers is used. The main contributions of this paper are: 1) synthesis w.r.t. a predefined periodic sampling time, rather than arbitrary switching with a (more conservative) minimum dwell-time and 2) the use of a different and less conservative CLBF. Additionally, more benchmark examples are provided. Other related work is found in [16], in which robust CLFs for switched systems with reach-while-stay (RWS) specifications are synthesized using a counterexample-guided synthesis. However, in [16] the controller modes are pre-specified, while in our approach these modes can also be discovered automatically, eliminating the need for prior input space discretization. Furthermore, this paper extends the set of specifications to include invariance of the goal set. Finally, similar to [15], the theoretical lower bounds on the minimum dwell-times reported in [16] are often very conservative.

a) Notation: Let $\mathbb{Z}_{\geq 0} = \{0, 1, 2 \dots\}$. Let us denote the boundary and the interior of a set D with ∂D and $int(D)$ respectively. The image and inverse image of set A under f are denoted by $f[A]$ and $f^{-1}[A]$. Finally, the Euclidean norm is denoted by $\|\cdot\|$.

II. PROBLEM DEFINITION

In this paper we design sampled-data state feedback controllers for nonlinear continuous-time systems described by

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{1}$$

This work is supported by NWO Domain TTW, the Netherlands, under the project CADUSY- TTW#13852.

C.F. Verdier and M. Mazo Jr are both with the Department of Delft Center of Systems and Control, Technical University of Delft, 2628 CD Delft, The Netherlands c.f.verdier@tudelft.nl

where the variables $\xi(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ denote the state and input respectively. Due to the sampled-data nature of the controller, $u(t) = g(x(t_k))$, $\forall t \in [t_k, t_k + h)$, where $h > 0$ denotes a constant sampling time.

A. Control specification

Given a compact safe set $S \subseteq \mathcal{X}$, compact initial set $I \subset S$ and compact goal set $G \subset S$, we consider the following specifications:

$\texttt{CS}_1$ *Reach while stay (RWS)*: all trajectories starting in I eventually reach G, while staying within S:

$$\forall \xi(t_0) \in I, \exists T, \forall t \in [t_0, T] : \xi(t) \in S \wedge \xi(T) \in G. \quad (2)$$

$\texttt{CS}_2$ *Reach and stay while stay (RSWS)*: all trajectories starting from I eventually reach and stay in G, while staying within S:

$$\forall \xi(t_0) \in I, \exists T, \forall t \geq t_0, \forall \tau \geq T : \xi(t) \in S \wedge \xi(\tau) \in G. \quad (3)$$

This paper addresses the following problem:

Problem 2.1: Given the compact sets (S, I, G) and system (1), synthesize a sampled-data state feedback controller $u(t) = g(x(t_k))$ such that the closed-loop system satisfies specification $\texttt{CS}_1$ or $\texttt{CS}_2$.

We propose to solve Problem 2.1 by using a periodically switched controller based on a CLBF, as will be established in the next section. The CLBFs and controller modes are synthesized using grammar-guided genetic programming (introduced in Section IV) and verified by means of an SMT solver. The overall algorithm is described in Section V.

III. CONTROL STRATEGY

In this section we discuss the used control strategy and establish how it solves problem 2.1 by means of Theorem 1 and Corollary 1.

A. Control Lyapunov Barrier Function

Consider a set of controller modes with index set $Q \subset \mathbb{Z}_{\geq 0}$:

$$\mathcal{G} = \{g_q : \mathcal{X} \to \mathcal{U} \mid q \in Q\}. \quad (4)$$

Given the system (1), an initial state $x = \xi(t_k)$, let us denote the (over-approximated) reachable set for $t \in [t_k, t_k + h]$ under a controller mode q as $R_q(x)$ s.t. given a q, $\forall t \in [t_k, t_k + h] : \xi(t) \in R_q(\xi(t_k))$. The construction of R_q is discussed in Section V-A. We consider a switching controller based on a CLBF defined as follows.

Definition 3.1 (Control Lyapunov Barrier Function): A function $V \in \mathcal{C}^1(S, \mathbb{R})$ is a Control Lyapunov Barrier Function (CLBF) w.r.t. the compact sets (S, I, G), $S \subseteq \mathcal{X}$, $I, G \subseteq int(S)$, system (1), and controller modes (4) if there exists a scalar $\gamma > 0$ such that

$$\forall x \in I : V(x) \leq 0 \quad (5a)$$

$$\forall x \in \partial S : V(x) > 0 \quad (5b)$$

$$\forall x \in A \backslash G, \exists q \in Q, \forall z \in R_q(x) : \dot{V}_q(x, z) \leq -\gamma \quad (5c)$$

where $A := \{x \in S \mid V(x) \leq 0\}$ and $\dot{V}_q(x, z) = \langle \nabla V(z), f(z, g_q(x)) \rangle$.

Remark 1: The choice of γ is arbitrary, because if a solution V^* exists for γ^*, there always exists a linear transformation of V^* such that the inequalities in (5) are satisfied for any γ.

Proposition 1: Given a CLBF V, $\exists e \in \mathbb{R}$ s.t. $e = \inf_{x \in S \backslash G} V(x)$. Furthermore, the sublevel set $L_c := \{x \in S \mid V(x) \leq c\}$ is compact.

Proof: Since $V(x)$ is continuous and S is compact, $V[S] \subset \mathbb{R}$ is compact and hence $V[S \backslash G] \subseteq V[S]$ is bounded, i.e. $\exists e \in \mathbb{R}$ s.t. $e = \inf_{x \in S \backslash G} V(x)$. Moreover, $Y := \{y \in V[S] \mid y \leq c\}$ and its inverse image $V^{-1}[Y] = L_c$ are compact. ∎

B. Control policy

Given a CLBF V, we consider periodically switching controllers of the form

$$\begin{cases} u(t) & = g_{q_k}(\xi(t_k)). \\ q_k(t_k) & = \arg\min\limits_{q \in Q} \max\limits_{z \in R_q(\xi(t_k))} \dot{V}_q(\xi(t_k), z) \end{cases} \quad (6)$$

where $t_{k+1} = t_k + h$, $t \in [t_k, t_k + h)$.

C. Reach while stay

The presented controller strategy based on the CLBF enforces specification $\texttt{CS}_1$, as shown in the following theorem.

Theorem 1: Given a system (1), CLBF V w.r.t. compact sets (S, I, G) and controller (6), then (2) holds.

Proof: For $\xi(t_0) \in I$ it follows from (5a) and the definition of A that $V(\xi(t_0)) \in A$. From (5c) it follows that for all $\xi(t_k) \in A \backslash G$ there exists a $q \in Q$ such that $\forall t \in [t_k, t_k + h] : \dot{V}_q(\xi(t_k), \xi(t)) \leq -\gamma$. Selecting such a mode using controller (6), applying the comparison theorem (see e.g. [17]), and using $\forall x \in A$, $V(x) \leq 0$, it follows that $\forall k \in \mathbb{Z}_{\geq 0}$, $\forall t \in [t_k, t_k + h]$, $\forall \xi(t_k) \in A \backslash G$: $V(\xi(t)) \leq V(\xi(t_k)) - \gamma h \leq -\gamma h$ Therefore, $\xi(t_k) \in A \backslash G$ implies $\forall t \in [t_k, t_k + h]$, $V(\xi(t))$ will decrease and thus cannot reach ∂S, as from (5b) we have $\forall x \in \partial S : V(x) > 0$. Since from proposition 1 it follows that $V(x)$ on $A \backslash G \subseteq S \backslash G$ is lower bounded, $V(\xi(t))$ will decrease until in finite time $\xi(t)$ leaves $A \backslash G$ and can only enter G, therefore (2) holds. ∎

D. Reach and stay while stay

The conditions in (5) are not sufficient for forward invariance of (a subset of) the goal set, as they do not impose that $\forall x \in \partial G, V(x) < \inf_{y \in S \backslash G} V(y)$. Therefore some trajectories starting in ∂G might enter $S \backslash G$ before entering G again. The following corollary establishes sufficient conditions for specification $\texttt{CS}_2$.

Corollary 1: Given a system (1), CLBF V w.r.t. compact sets (S, I, G), and a controller (6), if $\exists \beta \in \mathbb{R}$ such that

$$\forall x \in \partial G : \; V(x) > \beta \quad (7a)$$

$$\forall x \in G \backslash int(B), \exists q \in Q, \forall z \in R_q(x) : \dot{V}_q(x, z) \leq -\gamma \quad (7b)$$

where $B := \{x \in S \mid V(x) \leq \beta\}$, then (3) holds.

Proof: From Theorem 1 we have that there exists a time $t_K \geq t_0$ such that $\xi(t_K) \in G$. Analogous to the proof of Theorem 1, from (7b) it follows that $\forall \xi(t_K) \in G$, $\xi(t)$ with $t \geq t_K$ enters in finite time $G \cap B$. From the definition of B and Proposition 1 it follows that B is compact and thus $G \cap B$ is compact. From (7b) and controller (6) we have that

$\forall x \in \partial(G \cap B), z \in R_q(x) : \dot{V}_q(x,z) \leq -\gamma$. Combining this with (7a), we have that all states $\xi(t) \in \partial(G \cap B)$ cannot reach ∂G and $V(\xi(t))$ decreases, thus these trajectories will remain within $G \cap B$. Therefore it follows that $G \cap B \subseteq G$ is forward invariant. As $G \subseteq int(S)$, we have that (3) holds. ■

Remark 2: Comparing the CLBF for RSWS to the CLBF in [15], in this work the condition on the derivative of V is only imposed for the sublevel set $A \subset S$, rather than the entire safe set S. Secondly, the CLBF in this work involves only 2 parameters y and β, as opposed to 5 in [15].

IV. GRAMMAR-GUIDED GENETIC PROGRAMMING

Genetic programming (GP) is an evolutionary algorithm capable of synthesizing entire functions, in our case a CLBF and controller modes, that minimize a cost function, without pre-specifying a fixed structure [13]. The algorithm is initialized with a random population of candidate solutions (individuals). Each individual is assigned a fitness using the fitness function, which reflects how well the design goal is satisfied. Individuals are then selected based on this fitness to undergo genetic operations. The resulting individuals form a new generation. This cycle is repeated for many generations under the expectation that the average fitness increases, until a solution is found or a maximum number of generations is met. In GP, solutions (the phenotypes) are encoded in a certain representation (the genotypes) that allows for easy modification. In this work we use a grammar-guided genetic programming (GGGP) algorithm, similar to the work of [18], which uses a tree representation that is constructed based on a Backus-Naur form (BNF) grammar [19]. The BNF grammar consists of the tuple $\{\mathcal{N}, \mathcal{S}, \mathcal{P}, \mathcal{P}^*\}$, where $\mathcal{N}$ denotes the set of nonterminals, $\mathcal{S} \in \mathcal{N}$ the start symbol, $\mathcal{P}$ the set of production rules, and $\mathcal{P}^*$ the set of terminal production rules, which contains no recursive rules. An example of a simple grammar to construct monomials is given by $\mathcal{N} = \{\text{mon}, \text{var}\}$, $\mathcal{S} = \langle \text{mon} \rangle$, $\mathcal{P}$ in Table I, and $\mathcal{P}^*$ obtained by omitting the recursive rules from $\mathcal{P}$. Here $\langle \text{mon} \rangle$ denotes monomials and $\langle \text{var} \rangle$ scalar variables. Using $\mathcal{P}$, $\langle \text{mon} \rangle$ can be mapped to either $\langle \text{var} \rangle$ or $\langle \text{var} \rangle \times \langle \text{mon} \rangle$.

A parse tree is constructed using the BNF grammar as follows. Starting with the start symbol nonterminal $\mathcal{S}$, a random corresponding rule is chosen from the production rules $\mathcal{P}$. This rule forms a subtree that is put under the nonterminal. Subsequently, all nonterminals in the leaves of the resulting tree are similarly expanded, until all leave nodes contain no nonterminals anymore. To limit the tree depth, $\mathcal{P}^*$ is used if a predefined depth is reached, such that the number of recursive rules is limited. The final parse tree is transformed into the phenotype by replacing all nonterminals with their underlying subtrees, yielding a new parse tree corresponding directly to a function. Figure 1 shows a fully grown genotype synthesized using the example grammar, as well as the transformation to its phenotype.

We use the genetic operators crossover and mutation, which take the role of exploitation and exploration of genotypes respectively. The crossover operator takes two individuals and switches two random subtrees with the same nonterminal root. The mutation operator takes a single individual and replaces a subtree corresponding to a random nonterminal with a new subtree grown from that nonterminal.

TABLE I
PRODUCTION RULES $\mathcal{P}$

$\mathcal{N}$	Rules
$\langle \text{mon} \rangle$	$::= \langle \text{var} \rangle$
	$\mid \langle \text{var} \rangle \times \langle \text{mon} \rangle$
$\langle \text{var} \rangle$	$::= a \mid b$

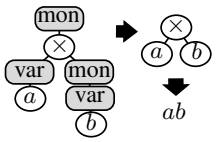

Fig. 1. Genotype to phenotype

As stated before, we aim to synthesize the pair $(V, \mathcal{G})$. For both the CLBF and controller modes we use a separate parse tree, which we refer to as a gene. In this case, the genotype is formed by two genes.

V. AUTOMATIC CLBF AND CONTROLLER SYNTHESIS

In this section the overall algorithm is described.

A. One-step ahead reachable set

In this work the reachable set is constructed by using Euler's forward method and bounding the local truncation error (LTE). This yields the following analytic expression

$$r_q(x, \tau, e) = x + \tau f(x, g_q(x)) + \frac{1}{2}\tau^2 e,$$

such that the over-approximated reachable set is given by $R_q(s) = \bigcup_{(\tau,e)\in E} r_q(s, \tau, e)$ with $E := [0, h] \times \Pi_{i=1}^{n}[-\varepsilon_i, \varepsilon_i]$ and

$$\varepsilon_i = \max_{(x,u)\in\mathcal{X}\times\mathcal{U}} \left| \frac{\partial f_i(x,u)}{\partial x} f_i(x,u) \right|. \tag{8}$$

While this construction can be quite conservative, it allows for relatively simple analytic expressions.

B. Fitness

For each inequality (5a)-(5c) (and optionally (7a)-(7b)), an independent fitness value is constructed, consisting of a sample-based and an SMT solver-based fitness value. The former gives a measure of how much the inequalities in (9) are violated, whereas the SMT solver is used to provide a formal guarantee on whether these inequalities are satisfied. In this work we use the SMT solver dReal [20], which is able to verify nonlinear inequalities over the reals. Furthermore, in case a formula is not satisfied, the SMT solver can be used to provide a counterexample, which can again be used for the sample-based verification.

Inequalities (5a)-(5c) and (7a)-(7b) can be rewritten as [1]

$$(\forall s \in C_i)\phi_i(s) \geq 0, \quad i = 1, \ldots, 5. \tag{9}$$

Given a finite set $C_{i,\text{samp}} = \{x_1, \ldots, x_n\}$, with $C_{i,\text{samp}} \subset C_i$, the sample-based fitness is based on an error measure w.r.t. ϕ_i defined as

$$e_{\phi_i} := \|[\min(0, \phi_i(x_1)), \ldots, \min(0, \phi_i(x_n)]\|.$$

[1]By taking $C_1 = I$, $C_2 = \partial S$, $C_3 = S \backslash G \times E$, $C_4 = \partial G$, $C_5 = G \times E$, $\phi_1(s) = -V(s)$, $\phi_2(s) = V(s) - c$, $\phi_3 = \chi_A(s)(-\gamma - \min_{q\in Q}(\dot{V}(s_1, r_q(s_1, s_2, s_3)), \phi_4(s) = V(s) - \beta - c$, $\phi_5 = \chi_B(s)(-\gamma - \min_{q\in Q}(\dot{V}(s_1, r_q(s_1, s_2, s_3)))$, where c is an arbitrary small real to make the strict inequality non-strict and $\chi_D(s)$ denotes a membership function of set D, i.e. $\chi_D(s) = 1$ if $s \in D$ and zero otherwise.

Using this measure, the sampled-based fitness is given by

$$f_{\text{samp},\phi_i} := (1 + e_{\phi_i})^{-1}, \quad i = 1, ..., 5. \tag{10}$$

The SMT-based fitness f_{SMT,ϕ_i} is 1 if it follows from dReal that the inequality is satisfied and 0 otherwise. This fitness value is only computed if an individual satisfies $f_{\text{samp},\phi_i} = 1$ for all $i \in \{1, 2, 3\}$ (or $i \in \{1, ..., 5\}$). Otherwise, f_{SMT,ϕ_i} are set to 0 for all conditions.

To prioritize finding a V that satisfies (5a) and (5b) before checking the condition on its derivative $\dot{V}$ in (5c), (and similarly for the additional conditions (7a)-(7b)), we use the weights $w_i = \lfloor f_{\text{samp},\phi_{i-1}} \rfloor$ for $i = 2, \dots 5$ and $w_1 = 1$. The overall fitness is given by

$$f := \sum_{i=1}^{j} w_i f_{\text{samp},\phi_i} + \sum_{i=1}^{j} f_{\text{SMT},\phi_i}, \ j = 3 \vee j = 5. \tag{11}$$

Finally, to promote the selection of equivalent, but less complex individuals, candidates with the same fitness (11) are ranked according to the number of their parameters. If this is still not decisive, they are subsequently ranked based on their lowest maximum parameter.

C. Numerical optimization

In order to speed up the convergence of the fitness, each generation the parameters of the individuals are optimized using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [21], which is an evolutionary optimization algorithm that is regarded to be robust with regard to discontinuous fitness functions. We use the variant sep-CMA-ES [22], because of its linear time and space complexity. In our grammar, we have the rule 'const' which creates a random constant. In every generation, these constants are then optimized using sep-CMA-ES, where their initial values are the current parameter values.

D. Algorithm outline

Provided a system, specification sets (S, I, G), a grammar, and sample sets $C_{i,\text{samp}}$, the proposed approach consists of the following steps:

1) A random population of individuals is generated as described in Section IV.
2) The parameters of the individuals are optimized using sep-CMA-ES based on the sample fitness.
3) The full fitness (11) is computed.
4) Counterexamples returned by the SMT solver are added to $C_{i,\text{samp}}$.
5) The best individuals are copied to the next generation.
6) A new population is created by selecting individuals using tournament selection [13] and modifying them using genetic operators.
7) Steps 3 to 5 are repeated until all conditions are satisfied (i.e. $f_{\text{SMT},\phi_i} = 1$ for all i) or a maximum number of generations is reached.

Remark 3: If the maximum number of generations is reached before an individual achieves $f_{\text{SMT},\phi_i} = 1$ for all i, no guarantees on the specification are provided.

Remark 4: It is possible to pre-define the controller modes in $\mathcal{G}$, such that only the CLBF is synthesized.

E. Additional operations

To aid finding the correct bias s of $V(x)$ such that (5a) is satisfied, the following biasing is performed before each fitness evaluation within CMA-ES:

$$V'(x) = V(x) - \max\Big(\max_{x \in I_{\text{samp}}}(V(x)), 0\Big), \tag{12}$$

where I_{samp} denotes a subsampled set of I. To guide the search further, we impose the additional condition

$$\forall x \in S \backslash G : V(x) \geq V(x_c) \tag{13}$$

where x_c denotes the center of the goal set.

VI. Implementation

The switching law in (6) is computationally intensive to check online. By offline designing $\alpha_q : \mathbb{R}^n \to \mathbb{R}$ for all $q \in Q$ such that

$$\begin{aligned} \forall x \in D, \forall q \in Q : \max_{z \in R_q(x)} \dot{V}_q(x, z) > -\gamma \implies \\ \min_{p \in Q}(\dot{V}_p(x, x) + \alpha_p(x)) < \dot{V}_q(x, x) + \alpha_q(x), \end{aligned} \tag{14}$$

allows us to replace the switching law with:

$$q_k(t_k) = \arg\min_{q \in Q}(\dot{V}_q(\xi(t_k), \xi(t_k)) + \alpha_q(\xi(t_k))). \tag{15}$$

Intuitively, when at a point x a mode q' is not viable under the reachable set $R_q(x)$, the nominal system $\dot{V}_{q'}(x, x)$ plus buffer $\alpha_{q'}(x)$ should not minimize the set $\bigcup_{q \in Q} \dot{V}_q(x, x) + \alpha_q(x)$, such that it is not selected by the switching law.

Theorem 2: Given a CLBF, if $\forall q \in Q$, $\alpha_q(x)$ satisfies (14) for $D = A \backslash G$, switching law (15) yields that (2) holds.

Proof: This proof is by contradiction. By definition of the CLBF, for all $x \in A \backslash G$, there always exists a q such that $\max_{z \in R_q(x)} \dot{V}_q(x, z) \leq -\gamma$. Assume that when using switching law (15), we have $\max_{z \in R_{q_k}(\xi(t))} \dot{V}_{q_k}(\xi(t), z) > -\gamma$. It then follows from (14) that $\min_{q \in Q}(\dot{V}_q(x, x) + \alpha_q(x)) < \dot{V}_{q_k}(x, x) + \alpha_{q_k}(x)$. This directly contradicts the switching law $q_k = \min_{q \in Q}(\dot{V}_q(x, x) + \alpha_q(x))$. Hence switching law (15) can only select a q_k such that $\max_{z \in R_{q_k}(\xi(t_k))} \dot{V}_{q_k}(\xi(t_k), z) \leq -\gamma$, which is guaranteed to exist by the design of the CLBF. The remainder of the proof is analogous to the proof of Theorem (1). ■

Corollary 2: Given a CLBF satisfying (7), if $\forall q$, $\alpha_q(x)$ satisfies (14) for $D = A \backslash int(B)$, using switching law (15) yields that (3) holds.

Proof: This proof is analogous to the proof of Theorem 2 and Corollary 1. ■

The functions $\alpha_q(x)$ can be designed and verified offline using again an SMT solver.

VII. Case studies

In this section the effectiveness of the approach is demonstrated for a simple linear system, polynomial systems of second and third order (see Table II), and two nonpolynomial systems (see Table III). The systems and specifications are adopted from [23] and references therein, with the exception of the Pendulum system, adopted from [15].

For these case studies, we fixed the control mode vector field $\mathcal{G}$ and synthesized controllers for the reach-while-stay

TABLE II

POLYNOMIAL SYSTEMS AND RESULTS FOR 10 RUNS. μ: MEAN, σ: STANDARD DEVIATION.

System	Linear				2nd-order				3rd-order			
Equations of motion	$\dot{x}_1 = x_2$ $\dot{x}_2 = -x_1 + u$				$\dot{x}_1 = x_2 - x_1^3$ $\dot{x}_2 = u$				$\dot{x}_1 = -10x_1 + 10x_2 + u$ $\dot{x}_2 = 28x_1 - x_2 - x_1x_3$ $\dot{x}_3 = x_1x_2 - 2.6667x_3$			
(S, I, G)	$([-1,\ 1]^2, [-0.5, 0.5]^2, [-0.1, 0.1]^2)$				$([-1,\ 1]^2, [-0.5, 0.5]^2, [-0.05,\ 0.05]^2)$				$([-5,\ 5]^3, [-1.2, 1.2]^3, [-0.3, 0.3]^3)$			
Predefined $\mathcal{G}$	$\{-1, 0, 1\}$				$\{-1, 0, 1\}$				$\{-100, -50, -5, 0, 5, 50, 100\}$			
ε, h	$(2, 1),\ 0.01$s				$(7, 0),\ 0.01$s				$(3800, 6800, 1900),\ 0.001$s			
	min	max	μ	σ	min	max	μ	σ	min	max	μ	σ
# generations	2	8	4.3	1.83	6	9	7.4	1.26	5	50	21.1	15
Total time [s]	7.34	35.09	15.69	8.53	32.17	59.50	47.15	10.24	37.65	475.307	197.99	125.32

TABLE III

NONPOLYNOMIAL SYSTEMS AND RESULTS FOR 10 RUNS. μ: MEAN, σ: STANDARD DEVIATION

System	Pendulum				Pendulum on a cart			
Equations of motion	$\dot{x}_1 = x_2$ $\dot{x}_2 = -\left(\frac{b}{J} + \frac{K^2}{JR_a}\right) x_2 - \frac{mlg}{J}\sin(x_1) + \frac{K}{JR_a}u$ $m = 5.50 \cdot 10^{-2}$ kg, $l = 4.20 \cdot 10^{-2}$ m, $J = 1.91 \cdot 10^{-4}$ kg m^2, $g = 9.81$ m/s^2, $b = 3.0 \cdot 10^{-6}$Nms, $K = 5.36 \cdot 10^{-2}$ Nm/A, $R_a = 9.50\ \Omega$.				$\dot{x}_1 = x_2$ $\dot{x}_2 = \frac{g}{l}\sin(x_1) - \frac{b}{ml^2}x_2 + \frac{1}{ml}\cos(x_1)u$ $g = 9.8$ m/s^2, $b = 2$ Nms $l = 0.5$ m, $m = 0.5$ kg.			
(S, I, G)	$([-2\pi,\ 2\pi] \times [-100,\ 100], [-\pi,\ \pi] \times [-10,\ 10],$ $[-1.0,\ -0.5] \times [-1.0,\ 1.0])$				$([-2\pi, 2\pi] \times [-10, 10], [-0.5, 0.5]^2,$ $[-0.25, 0.25] \times [-1, 1])$			
Predefined $\mathcal{G}$	$\{-10, -5, 0, 5, 10\}$				$\{-6, -2, 0, 2, 6\}$			
$\varepsilon,\ h$	$(600, 12700),\ 0.001$s				$(200, 3200),\ 0.01$s			
	min	max	μ	σ	min	max	μ	σ
# generations	4	20	9.3	4.52	4	15	8.6	3.13
Total time [s]	22.79	107.25	63.61	26.13	37.16	114.492	88.34	25.32

specification $\texttt{CS}_1$. Across all these case studies, we used a population of 16 individuals, a maximum of 50 generations, and a maximum of 30 generations within CMA-ES. The mutation and crossover rates were both chosen to be 0.5. The number of test samples and maximum number of additional counterexamples were set to 100 and 300 respectively. For the counterexamples, a first-in-first-out principle was used. The (arbitrary) γ of the CLBF was set to $\gamma = 0.1$ and the precision parameter of dReal set to $\delta = 0.001$. The values of ε_i are obtained using bisection and the SMT solver. The GGGP algorithm and CMA-ES are implemented in Mathematica, running on an Intel Xeon CPU E5-1660 v3 3.00GHz using 8 CPU cores.

The used grammar is defined by $\mathcal{S}_V = \langle \text{const} \rangle + \langle \text{expr} \rangle$, $\mathcal{N}$ and $\mathcal{P}$ as shown in Table IV, and $\mathcal{P}^*$ is obtained by removing all recursive rules from $\mathcal{P}$. While this grammar restricts to polynomial CLBFs, the proposed approach can also be used for nonpolynomial CLBFs. Finally, the maximum recursive rule depth was set to be 7.

To show repeatability, the synthesis was repeated 10 times for each benchmark. Statistics on the number of generations and the total synthesis time are shown in Table II and III. With the exception of the third-order polynomial system, in all 10 runs a solution was found for each benchmark. For the third-order system only a single run did not find a solution within 50 generations.

TABLE IV

PRODUCTION RULES $\mathcal{P}$

$\mathcal{N}$	Rules
$\langle \text{expr} \rangle$	$::= \langle \text{expr} \rangle + \langle \text{expr} \rangle \mid \langle \text{pol} \rangle$
$\langle \text{pol} \rangle$	$::= \langle \text{pol} \rangle + \langle \text{pol} \rangle \mid \langle \text{const} \rangle \times \langle \text{mon} \rangle$
$\langle \text{mon} \rangle$	$::= \langle \text{var} \rangle \mid \langle \text{var} \rangle \times \langle \text{var} \rangle$
$\langle \text{var} \rangle$	$::= x_1 - x_{c,1} \mid \ldots \mid x_n - x_{c,n}$
$\langle \text{const} \rangle$	::= Random Real $\in [-10, 10]$
$\langle \text{G} \rangle$	$::= \{\langle \text{lin} \rangle\} \mid \ldots \mid \{\langle \text{lin} \rangle, \langle \text{lin} \rangle, \langle \text{lin} \rangle\},$
$\langle \text{lin} \rangle$	$::= \langle \text{const} \rangle (x_1 - x_{c,1}) + \cdots + \langle \text{const} \rangle (x_n - x_{c,n})$ $\mid \langle \text{const} \rangle \langle \text{var} \rangle \mid \langle \text{const} \rangle$

One of the found solutions for the pendulum system is

$$V = -2319.91 + 102.46{x'_1}^2 + 8.88356x'_1x_2 + 1.54932x_2^2,$$

where $x'_1 = (0.75 + x_1)$. We manually designed $\alpha(x) = [\alpha_1(x), \ldots \alpha_5(x)]^T$ to be $[100, 0, 0, 0, 400]$ such that (14) holds. Furthermore, by post-analyzing V, it was proven that for $\beta = -2304.69$, V satisfies the conditions in (7), hence the stronger specification $\texttt{CS}_2$ is guaranteed. Figure 2 shows the phase plot of the closed-loop system for $\xi(t_0) \in \{(-\pi, 10), (-2, -5), (1.5, 0), (\pi, 10)\}$. It can be seen that indeed all trajectories satisfy $\texttt{CS}_2$.

The used sampling times h are significant larger than the minimal dwell-times reported in [15] and [16]. For example, for the cart on pendulum system benchmark we used $h = 0.01$ s, whereas [16] reports a theoretical minimum dwell-time of $2 \cdot 10^{-6}$ s with an observed minimum switch time of 0.005 seconds in simulation experiments.

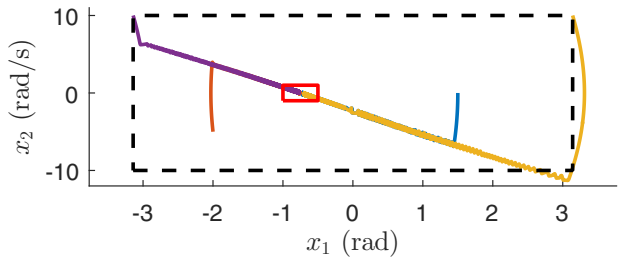

Fig. 2. Phase diagram of the pendulum system using a found CLBF.

TABLE V

RESULTS FOR 10 RUNS FOR THE PENDULUM ON A CART SYSTEM WITHOUT PRE-DEFINING $\mathcal{G}$.

	Min	Max	μ	σ
# generations	3	15	7.6	3.37
Total time [s]	84.58	349.80	219.29	86.11

A. Evolving $\mathcal{G}$

Let us reconsider the pendulum on a cart from Table III and specification $\texttt{CS}_1$, but without pre-specifying $\mathcal{G}$. We saturate the input with $u(t_k) = \max(-6, \min(6, g_{q_k}(\xi(t_k))))$. A separate gene for the controller modes $\mathcal{G}$ is used with start symbol $\mathcal{S}_G = \langle \mathrm{G} \rangle$ and the product rules in Table IV. The results for 10 runs are shown in Table V. Comparing Table III with V we observe a comparable number of generations required to find a solution, although a longer computation time per generation is observed. However, the benefit is that no discretization of the input space is required. One of the found solutions is given by

$$(V, \mathcal{G}) = (100.459x_1^2 + 36.111x_1x_2 + 22.7543x_2^2 - 40.3802,\ \{-10.0347x_1\}).$$

Note that $\mathcal{G}$ consists of only a single mode, hence no switching law is required when implementing this controller. Finally, for $\beta = -36.6211$, V satisfies (7), hence using this controller also guarantees $\texttt{CS}_2$.

VIII. DISCUSSION

This paper presented a method for automatic synthesis of a periodically switched state feedback controller for nonlinear sampled-data systems with reachability and safety specifications. Preliminary results have been shown for several nonlinear systems up to the third-order. It was shown that the framework was able to synthesize CLBFs given pre-defined controller modes, but was also capable of synthesizing controller modes automatically, eliminating the need to discretize the input space beforehand. Moreover, it is possible to find a single controller automatically, removing the need for the switching law entirely.

For almost all benchmarks runs, solutions were found within 50 generations. Nevertheless, a drawback of the proposed methodology is that there is no guarantee that a solution will be found within a number of generations.

A straightforward improvement to obtain less conservative sampling times is by using a less conservative over-approximation of the reachable set, e.g. by using higher order Taylor series approximations or using local bounds rather than for the entire domain.

Finally, the functions $\alpha_q(x)$ that simplify the switching condition are currently synthesized by hand. In future work, the aim is to automate this synthesis as well, for example by again using the combination of GP with SMT solvers.

REFERENCES

[1] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer US, 2009.

[2] M. Mazo, A. Davitian, and P. Tabuada, "Pessoa: A tool for embedded controller synthesis," in *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 566–569.

[3] M. Rungger and M. Zamani, "Scots: A tool for the synthesis of symbolic controllers," in *Proc. of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 99–104.

[4] S. Mouelhi, A. Girard, and G. Gössler, "Cosyma: a tool for controller synthesis using multi-scale abstractions," in *Proc. of the 16th international conference on Hybrid systems: computation and control*. ACM, 2013, pp. 83–88.

[5] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 7, no. 11, pp. 1163 – 1173, 1983.

[6] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *Proc. of the 7th IFAC Symposium on Nonlinear Control Systems*, pp. 462–467, 2007.

[7] M. Z. Romdlony and B. Jayawardhana, "Stabilization with guaranteed safety using control lyapunovbarrier function," *Automatica*, vol. 66, pp. 39 – 47, 2016.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2016.

[9] A. Papachristodoulou and S. Prajna, "On the construction of lyapunov functions using the sum of squares decomposition," in *Decision and Control, 2002, Proc. of the 41st IEEE Conference on*, vol. 3, Dec 2002, pp. 3482–3487 vol.3.

[10] W. Tan and A. Packard, "Searching for control lyapunov functions using sums of squares programming," in *Allerton Conferencei*, 2004, pp. 210–219.

[11] E. J. Hancock and A. Papachristodoulou, "Generalised absolute stability and sum of squares," *Automatica*, vol. 49, no. 4, pp. 960 – 967, 2013.

[12] A. A. Ahmadi, M. Krstic, and P. A. Parrilo, "A globally asymptotically stable polynomial vector field with no polynomial lyapunov function." in *CDC-ECE*, 2011, pp. 7579–7580.

[13] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[14] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories." *Handbook of satisfiability*, vol. 185, pp. 825–885, 2009.

[15] C. Verdier and J. M. Mazo, "Formal controller synthesis via genetic programming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7205 – 7210, 2017, 20th IFAC World Congress.

[16] H. Ravanbakhsh and S. Sankaranarayanan, "Robust controller synthesis of switched systems using counterexample guided framework," in *Embedded Software (EMSOFT), 2016 International Conference on*. IEEE, 2016, pp. 1–10.

[17] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.

[18] P. A. Whigham *et al.*, "Grammatically-based genetic programming," in *Proc. of the workshop on genetic programming: from theory to real-world applications*, 1995, pp. 33–41.

[19] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger, "Revised report on the algorithm language algol 60," *Commun. ACM*, vol. 6, no. 1, pp. 1–17, Jan. 1963.

[20] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *International Conference on Automated Deduction*. Springer, 2013, pp. 208–214.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] R. Ros and N. Hansen, *A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 296–305.

[23] H. Ravanbakhsh and S. Sankaranarayanan, "Counterexample guided synthesis of switched controllers for reach-while-stay properties," *arXiv preprint arXiv:1505.01180*, 2015.