Delft University of Technology

# Formal controller synthesis via genetic programming

Verdier, C. F.; Mazo, M.

**DOI**
10.1016/j.ifacol.2017.08.1362

**Publication date**
2017

**Document Version**
Final published version

**Published in**
IFAC-PapersOnline

**Citation (APA)**
Verdier, C. F., & Mazo, M. (2017). Formal controller synthesis via genetic programming. *IFAC-PapersOnline*, *50*(1), 7205-7210. https://doi.org/10.1016/j.ifacol.2017.08.1362

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

**Copyright**
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*
*For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.*

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 7205–7210

# Formal Controller Synthesis via Genetic Programming

C.F. Verdier [*] M. Mazo, Jr. [*]

[*] Department of Delft Center of Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: {c.f.verdier, m.mazo}@tudelft.nl).

**Abstract:** This paper presents an automatic controller synthesis method for nonlinear systems with reachability and safety specifications. The proposed method consists of genetic programming in combination with an SMT solver, which are used to synthesize both a control Lyapunov function and the modes of a switched state feedback controller. The resulting controller consists of a set of analytic expressions and a switching law based on the control Lyapunov function, which together guarantee the imposed specifications. The effectiveness of the proposed approach is shown on a 2D pendulum.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

*Keywords:* Formal methods, Lyapunov methods, Genetic programming

## 1. INTRODUCTION

Complex controller specifications of modern cyber-physical systems can often be formulated as (linear) temporal properties, i.e. propositions that are qualified in terms of time. In this paper we limit ourselves to a very specific temporal property, i.e. the *reach and stay while stay* (RSWS) property: all trajectories starting in an initial set $I$ *eventually* reach and stay within the goal set $G$, while *always* staying in the safe set $S$. The aim of this paper is to automate the synthesis of switched state feedback controllers for nonlinear systems, such that the RSWS specification is met.

Two popular correct-by-design controller synthesis approaches for nonlinear systems with temporal specifications are 1) abstraction and simulation, and 2) control Lyapunov functions (CLFs) and control barrier functions (CBFs). The first approach consists of finding a symbolic abstraction of the system that (bi)simulates the original system, for which it is easier to construct and verify a controller to satisfy temporal logic specifications (Tabuada (2009)). Tools implementing this methodology for nonlinear systems are e.g. PESSOA (Mazo Jr et al. (2010)), SCOTS (Rungger and Zamani (2016)), and CoSyMa (Mouelhi et al. (2013)). Drawbacks of these methods are that they require a discretization of the state space and the controllers often take the form of enormous tables, hence these methods suffer from the curse of dimensionality.

Control Lyapunov functions (CLF) (Artstein (1983)) and control barrier functions (CBF) (Wieland and Allgöwer (2007)) are design tools for stabilization and safety respectively. The benefit of these methods is that there is no need to compute the exact solution of the system. Attempts to unify CLFs with CBFs can be found in e.g. Romdlony and Jayawardhana (2016) and Xu et al. (2015). A popular approach to automatic synthesis of (control) Lyapunov functions and (control) barrier functions is to pose the problem as a sum of squares (SOS) problem, which reduces it to a convex optimization problem, see e.g. Papachristodoulou and Prajna (2002). This approach is restricted to polynomial systems, although extensions exist such that some non-polynomial functions can be used by recasting them as polynomials, see e.g. Chesi (2009) and Hancock and Papachristodoulou (2013). Nevertheless, polynomial Lyapunov functions can be too restrictive. As shown in Ahmadi et al. (2011), if a polynomial system is globally asymptotically stable, there might exist no polynomial Lyapunov function.

To overcome the limitations of the two discussed approaches, we propose to use genetic programming (GP). GP is an evolutionary algorithm capable of evolving encoded representations of symbolic functions, until a satisfactory solution is found (Koza (1992)). The evolution is driven by a fitness function, which scores solutions on how well they satisfy desired specifications. GP distinguishes itself from other optimization methods in that it is able to search over the function space, rather than over a parameter space. Due to this nature, genetic programming (and variants) have been used to synthesize Lyapunov functions, see e.g. Grosman and Lewin (2009), and controllers, see e.g. Koza et al. (2003), Sekaj and Perkacz. (2007), Diveev and Shmalko (2015), and Chen and Lu (2011). In these works, fitness is based on specific samples and/or simulations, hence no formal guarantee can be given on the behavior of the system, other than for the specific test cases. In this work we propose the combination of genetic programming and a Satisfiability Modulo Theories (SMT) solver (Barrett et al. (2009)), which uses a combination of background theories to determine whether a first-order logic formula can be satisfied or not. This solver is used to provide formal guarantees on the behavior of the system.

In our approach, the used control strategy is a switching law that switches between different controller modes based on a CLF. Genetic programming is used to automatically

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

2405-8963 © 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.1362

generate both candidate CLFs and the controller modes. Subsequently, the candidate solutions are verified using the SMT solver. In this paper, the SMT solver dReal is used, which is capable of providing formal guarantees on the satisfiability of nonlinear inequalities over the real numbers (Gao et al. (2013)). By using GP, we allow ourselves to search for solutions that include non-polynomial functions. Furthermore, as opposed to abstraction methods, the synthesized controllers are expressed as analytic expressions, that are in general more compact than a binary decision diagram (BDD) or a lookup table. Finally, the proposed method provides formal guarantees on stability and safety, as opposed to previous attempts using GP.

A similar approach is found in Ravanbakhsh and Sankaranarayanan (2015). Here, counter-example guided synthesis is used to synthesize a CLF for a switched system with a *reach-while-stay* specification. The verification is also done using dReal. However, only the controller modes are prefixed and only the CLF is synthesized.

To the best knowledge of the authors, this is the first work combining genetic programming and formal verification for controller synthesis. Furthermore, a special CLF is designed for the RSWS specification and such that the verification is decidable.

## 2. PROBLEM DEFINITION

**Notation:** Given a set $A$, let us denote the boundary as $\partial A$ and the interior as $int(A)$. The Euclidean norm is denoted by $\|\cdot\|$ and the natural logarithm by $\ln(\cdot)$. The temporal logic operators *always* and *eventually* are denoted by $\Box$ and $\Diamond$ respectively. The predicate defining set $A$ is denoted by $\phi_A$. A system satisfies $\Box\phi_A$ if and only if $\forall t \geq 0, \xi(t) \in A$ and a system satisfies $\Diamond\phi_A$ if and only if $\exists t \geq 0, \xi(t) \in A$.

In this work we consider the class of nonlinear dynamical systems described by

$$\begin{cases} \dot{\xi}(t) &= f(\xi(t), u(t)) \\ \xi(0) &\in I \end{cases} \tag{1}$$

where $I$ is a compact set and the variables $\xi(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ denote the state and input respectively.

The controller is designed such that the composition of the system (1) and control input $u(t)$ for $t \geq 0$ results in a system that satisfies

S1) *Reach and stay while stay (RSWS)*: given a compact safe set $S$, all trajectories starting from the compact initial set $I \subset int(S)$ eventually reach and stay in the compact goal set $G \subset int(S)$, while staying within the safe set $S$. This corresponds to the temporal logic formula $\Box\phi_S \wedge \Diamond\Box\phi_G$.

S2) There occurs no Zeno behavior.

We say a there occurs no Zeno behavior if there are no infinitely many switches in a finite time interval.

This paper addresses the following problem:

*Problem 1.* Given the compact sets $S$, $I$, $G$ and system (1), synthesize a control law $u(t)$ such that specifications S1) and S2) are guaranteed.

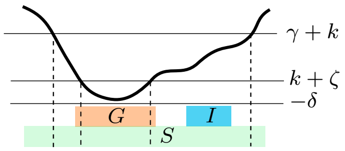

Fig. 1. Example of a RSWS CLF.

## 3. CONTROL STRATEGY

Consider the index set $Q = \{1, \ldots, M\}$, controller mode $q \in Q$ and vector fields $H = \{h_q(\xi(t)) | q \in Q\}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. In this work, we consider controllers of the form

$$u(t) = h_q(\xi(t)). \tag{2}$$

A switching law based on a CLF determines the next mode $q(\xi(t^+))$. The CLF is designed such that in combination with the switching law the desired safety and reachability specifications of S1) are enforced. This specific CLF is referred to as the RSWS CLF.

### *3.1 RSWS control Lyapunov function*

Classical (control) Lyapunov functions satisfy a "tight" inequality, i.e. $(\forall x \in D)V(x) \geq 0$ and $\exists x \in D$ such that $V(x) = 0$, where $\{0\} \subseteq D$, see e.g. Khalil (2002). Similarly, the first derivative also satisfies a tight inequality. As stated in Gao et al. (2012), satisfiability of inequalities over the reals for transcendental functions is not decidable, hence they introduced the $\delta$-decision problem, which is decidable. However, the $\delta$-decision problem can be problematic for tight inequalities, as will be shown in Section 5. To circumvent the occurrence of tight inequalities, we introduce a perturbation variable $\delta$ in our definition of the RSWS CLF, yielding a more general CLF-like function, coined the *relaxed RSWS CLF*. The term relaxed is used to indicate that the bounds are picked to be more conservative compared to the nominal RSWS CLF (i.e. $\delta = 0$).

Let us denote the Lie derivative of $g(x)$ along the flow of $f(x, h_q(x))$ as $L_{f_q} = \frac{\partial g(x)}{\partial x} f(x, h_q(x)))$.

*Definition 2.* (Relaxed RSWS control Lyapunov function). A function $V \in \mathcal{C}^2(S, \mathbb{R})$ is a relaxed RSWS control Lyapunov function w.r.t. the compact sets $(S, I, G)$ and system (1), if there exists real numbers $\alpha, \beta, \gamma, \varepsilon, \zeta > 0$, and $\delta \geq 0$, such that

$$\begin{array}{ll} V(x) \geq k + \zeta & \forall x \in S \backslash G \\ V(x) \geq -\delta & \forall x \in G \\ V(x) > \gamma + k & \forall x \in \partial S \\ V(x) \leq \gamma + k & \forall x \in I \\ \exists q \in Q \text{ x.t. } \dot{V}_q(x) \leq -\alpha V(x) + \delta & \forall x \in S \\ \ddot{V}_q(x) \leq \varepsilon & \forall q \in Q, \forall x \in S \end{array} \tag{3}$$

where $\dot{V}_q(x) = L_{f_q} V$, $\ddot{V}_q(x) = L_{f_q} \dot{V}_q(x)$ and

$$k = \beta + \frac{\delta}{\alpha}. \tag{4}$$

*Remark 3.* Note that the relaxed control Lyapunov function is not a CLF in the strict sense if $\delta > 0$, as it and its time derivative are not necessarily positive definite and negative definite respectively.

To illustrate the first four conditions, an example RSWS CLF is shown in Figure 1. For the sake of brevity, we use CLF to refer to the relaxed RSWS CLF throughout this paper.

### *3.2 Switching law*

Let us consider the switching law:

$$q(t^+) := \begin{cases} \arg\min_{q\in Q} \dot{V}_q(\xi(t)) \text{ if } \dot{V}_{q(t)}(\xi(t)) \geq \\ \qquad\qquad\qquad -\alpha V(\xi(t)) + \alpha\beta + \delta \\ q(t) \qquad\qquad\quad \text{otherwise} \end{cases} \tag{5}$$

where $q(t) \in Q$ and $q(t^+) \in Q$ denote the current and next controller mode respectively. As shown in the next lemma, this switching law in combination with a CLF in (3) results in a minimal dwell time between controller modes, i.e. the controller mode can only switch after a fixed nonzero period of time.

*Lemma 4.* Given the switching law in (5) and a function $V \in \mathcal{C}^2(S,\mathbb{R})$ satisfying (3), then for each $q(t) \in Q$, $\xi(t) \in S$ and $0 \leq t_i \leq t_{i+1}$ s.t. $\dot{V}_q(\xi(t_i)) \leq -\alpha V(\xi(t_i)) + \delta$ and $\dot{V}_q(\xi(t_{i+1})) = -\alpha V(\xi(t_{i+1})) + \alpha\beta + \delta$, there exists a minimal dwell time $\tau_d > 0$ such that $t_{i+1} - t_i \geq \tau_d$.

**Proof.** Without loss of generality, let us consider a controller mode $q^*$, initial time $t_i = 0$ and the first time at which a switch would occur $t_{i+1} = t_i + \tau_d$, such that

$$\dot{V}_{q^*}(\xi(0)) \leq -\alpha V(\xi(0)) + \delta, \tag{6}$$

$$\dot{V}_{q^*}(\xi(\tau_d)) = -\alpha V(\xi(\tau_d)) + \alpha\beta + \delta. \tag{7}$$

Consider the following equality:

$$\dot{V}_{q^*}(\xi(\tau_d)) = \dot{V}_{q^*}(\xi(0)) + \int_0^{\tau_d} \ddot{V}_{q^*}(\xi(\tau))d\tau. \tag{8}$$

Substituting (6) and (7) into (8) and using the bound on $\ddot{V}_{q^*}(x)$ from (3) yields:

$$-\alpha V(\xi(\tau_d)) + \alpha\beta \leq -\alpha V(\xi(0)) + \varepsilon\tau_d. \tag{9}$$

Furthermore, we have for all $t \in [0, \tau_d]$ that

$$\dot{V}_{q^*}(\xi(t)) \leq -\alpha V(\xi(t)) + \alpha\beta + \delta \leq \Delta, \tag{10}$$

with $\Delta = \alpha\delta + \alpha\beta + \delta$. It directly follows that

$$V(\xi(\tau_d)) = V(\xi(0)) + \int_0^{\tau_d} \dot{V}_{q^*}(\xi(\tau))d\tau \leq V(\xi(0)) + \Delta\tau_d,$$

and thus $-\alpha V(\xi(\tau_d)) \geq -\alpha V(\xi(0)) - \alpha\Delta\tau_d$. Substituting this in (9) yields

$$-\alpha V(\xi(0)) - \alpha\Delta\tau_d + \alpha\beta \leq -\alpha V(\xi(0)) + \varepsilon\tau_d.$$

Therefore, we obtain

$$\tau_d \geq \frac{\alpha\beta}{\varepsilon + \alpha(\alpha\delta + \alpha\beta + \delta)}, \tag{11}$$

with real numbers $\alpha, \beta, \gamma, \varepsilon > 0, \delta \geq 0$ and $\delta \geq 0$. Hence it follows that there exists a minimal dwell time $\tau_d > 0$ .

### *3.3 Closed-loop system*

The interconnection of system (1) and the control law (2), using the switching law in (5) and the CLF in (3), yields a closed-loop system with the properties stated in the following theorem.

*Theorem 5.* Given the compact sets $S, I \subset int(S), G \subset int(S)$, system (1), controller (2), switching law (5), and a relaxed CLF satisfying (3), the closed-loop system satisfies specifications S1) and S2).

**Proof.** From Lemma 4 it directly follows that for switching law (5) and a CLF satisfying (3), there exist a minimum dwell time strictly larger than zero, hence there are not infinitely many switches in a finite interval and therefore S2) is proven.

The initial conditions of the system satisfy $\xi(0) \in I$, where $I \subset int(S)$. From (3) it follows that

$$V(\xi(0)) \leq \gamma + k, \quad \forall \xi(0) \in I. \tag{12}$$

Let us consider a controller mode sequence by $Q^* := \{q_k\}_{k\in\mathbb{N}}$, found by applying the switching law (5). From (3) and (5) we have for all $q^* \in Q^*$

$$\dot{V}_{q^*}(\xi(t)) \leq -\alpha V(\xi(t)) + \alpha\beta + \delta, \ \forall \xi(t) \in S. \tag{13}$$

Using the comparison lemma (see e.g. Khalil (2002)) yields

$$V(\xi(t)) \leq (V(\xi(0)) - k)e^{-\alpha t} + k, \quad \forall \xi(t) \in S, \tag{14}$$

where $k$ is defined in (4). From (12) it follows that

$$(V(\xi(0)) - k)e^{-\alpha t} + k \leq \gamma e^{-\alpha t} + k, \ \ \forall \xi(0) \in I.$$

Substituting this in (14) yields that for all initial conditions $\xi(0) \in I$ and $t \geq 0$ we have

$$V(\xi(t)) \leq \gamma e^{-\alpha t} + k \leq \gamma + k, \quad \forall \xi(t) \in S. \tag{15}$$

Hence, given $\xi(0) \in I$, where $I \subset int(S)$, we have $V(\xi(t)) \leq \gamma + k$ while $\xi(t) \in S$. Since for all $x \in \partial S$, $V(x) > \gamma + k$, we have that for all $\xi(0) \in I$, $\xi(t)$ remains within the interior of $S$ for all $t \geq 0$, hence $\square\phi_S$ is proven.

To show that eventually all trajectories starting in $I$ reach and stay in set $G$, we use the previously found properties that for all $\xi(0) \in I$ and all $t \geq 0$, we have $\xi(t) \in S$ and $V(\xi(t))$ is bounded by $V(\xi(t)) \leq \gamma e^{-\alpha t} + k$. From (3) we have $V(x) \geq k + \zeta$ for all $x \in S\backslash G$, hence for all $t > -\frac{1}{\alpha}\ln(\frac{\zeta}{\gamma})$ we have $\xi(t) \in G$. Therefore all trajectories satisfying $\xi(0) \in I$ eventually converge and stay in $G$, i.e. $\Diamond\square\phi_G$ is satisfied.

Now the objective is to find the parameters $\alpha, \beta, \gamma, \zeta, \varepsilon$ and synthesize both $V(x)$ and the vector fields $H$ such that $V(x)$ is a CLF, i.e. the conditions in (3) hold.

## 4. GRAMMAR-GUIDED GENETIC PROGRAMMING

The classical GP algorithm as introduced in Koza (1992) consists of a population of functions encoded as parse trees (the genotypes). Based on a fitness function, each individual within the population is assigned a fitness. With a probability related to the fitness, solutions are selected to reproduce and undergo genetic operators to form new individuals for a new population. By repeating this for many generations, the underlying assumption is that the average fitness of the population increases, ultimately finding a suitable solution.

In this work we use a variant of GP, namely grammar-guided genetic programming (GGGP), similar to the one proposed in Whigham et al. (1995). In GGGP, parse trees are synthesized based on a user-defined grammar. We use a grammar in Backus-Naur form (BNF) (Naur (1963)), consisting of the tuple $\{\mathcal{N}, \mathcal{S}, \mathcal{P}, \mathcal{P}^*\}$. Here $\mathcal{N}$ presents the set of nonterminals, $\mathcal{S} \in N$ the start symbol, $\mathcal{P}$ the set of production rules, and $\mathcal{P}^*$ the set of terminal production rules, which does not contain recursive rules.

*Example 6.* An example of a simple grammar consisting of nonterminals for expressions $\langle\text{expr}\rangle$, monomials $\langle\text{mon}\rangle$, and variables $\langle\text{var}\rangle$ is given by $\mathcal{N} = \{\text{expr}, \text{mon}, \text{var}\}$, $\mathcal{S} = \langle\text{expr}\rangle$ and the production rules $\mathcal{P}$ in Table 1. The

Table 1. Production rules $\mathcal{P}$.

| Nonterminal | Production rules |
|---|---|
| $\langle\text{expr}\rangle$ ::= | $\langle\text{mon}\rangle$ |
| | $\mid\ \langle\text{expr}\rangle + \langle\text{mon}\rangle$ |
| $\langle\text{mon}\rangle$ ::= | $\langle\text{var}\rangle$ |
| | $\mid\ \langle\text{var}\rangle \times \langle\text{mon}\rangle$ |
| $\langle\text{var}\rangle$ ::= | $a \mid b$ |

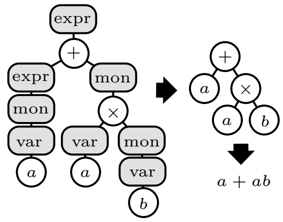

Fig. 2. Transformation.

terminal rules $\mathcal{P}^*$ are obtained by removing the recursive rules from $\mathcal{P}$.

The production rules are used to map nonterminals to a number of productions that are delimited by '|'. For example, in Table 1, the nonterminal $\langle\text{expr}\rangle$ can be mapped to either the nonterminal $\langle\text{mon}\rangle$ or the sum of the nonterminals $\langle\text{expr}\rangle$ and $\langle\text{mon}\rangle$.

A parse tree is synthesized using the BNF grammar as follows. The tree is initialized with the starting symbol. Based on this nonterminal, a random corresponding rule is picked from $\mathcal{P}$. The subtree corresponding to the rule is put under the starting symbol, forming a new tree. Nonterminals at the leaf nodes are subsequently expanded similar to the starting symbol, until all leaf nodes do not contain nonterminals. If a predefined depth is reached, the terminal production rules $\mathcal{P}^*$ are used to expand nonterminals, preventing an infinite depth tree. To translate genotype into a function, all nonterminals are removed by replacing them with their underlying subtrees, resulting in a new parse tree that can be expressed as a function. Going back to Example 6, a fully grown genotype based on this grammar and the transformation to the corresponding function is shown in Figure 2.

Two popular genetic operators in classical GP are crossover and mutation. In crossover, two random subtrees of two individuals are interchanged. In mutation, a random subtree of an individual is replaced by a new randomly generated subtree. In GGGP the genetic operators crossover and mutation are very similar. Here, for crossover and mutation a random nonterminal is picked and the branch underneath is replaced by another tree corresponding to the same type of nonterminal.

## 5. AUTOMATIC SYNTHESIS

The inequalities of (3) can be put in the standard form

$$(\forall x \in C_i)\phi_i(x) \geq 0, \quad i = 1, \ldots, 6 \tag{16}$$

with

$$\begin{aligned}
\phi_1 &= V(x) - k - \zeta \\
\phi_2 &= V(x) + \delta \\
\phi_3 &= V(x) - \gamma - k + c \\
\phi_4 &= -V(x) + \gamma + k \\
\phi_5 &= -\min_{q \in Q} \dot{V}_q(x) - \alpha V(x) + \delta \\
\phi_6 &= -\max_{q \in Q} \ddot{V}_q(x) + \varepsilon
\end{aligned} \tag{17}$$

$C_1 = S \backslash G$, $C_2 = \ G$, $C_3 = \partial S$, $C_4 = I$, $C_5 = C_6 = S$, and $c > 0$ is an arbitrary small number such that $\phi_i(x) - c \geq 0 \implies \phi_i(x) > 0$. Given a function $V$, controller $u(x, q)$, and the system $f(x, u(x, q))$, the conditions in (16) are used to verify whether $V$ is a relaxed RSWS CLF. In this work we use an approximation of the satisfiability based on an error over a finite number of points in $C_i$ and a formal verification based on an SMT solver. Here the latter gives a binary answer on the satisfiability and the former an indication how close an unsatisfied inequality is to being satisfied.

Given a vector $x_{t,i} = [x_1, \ldots, x_n]$ where $x_j \in C_i$ for $j = 1, \ldots, n$, we define the error measure w.r.t the inequality $\phi_i(x) \geq 0$ as

$$e_{\phi_i}(x_{t,i}) = \| \, [\min(0, \phi_i(x_1)), \ldots, \min(0, \phi_i(x_n))] \, \|. \tag{18}$$

### 5.1 SMT solver

We prove a logic formula $\psi_i := (\forall x \in C_i)\phi_i(x) \geq 0$ is satisfied, by proving that $\neg\psi_i \equiv (\exists x \in C)\phi_i(x) < 0$ is unsatisfied. As shown in Gao et al. (2012), these formulae are equivalent to

$$\varphi := (\exists z \in Z) \left( \bigwedge\nolimits_{i=1}^{m} \left( \bigvee\nolimits_{j=1}^{k_i} g_{ij}(x) = 0 \right) \right). \tag{19}$$

In this work we use the SMT solver dReal, which implements the $\delta$-decision problem for nonlinear functions over the real numbers, i.e. deciding whether a formula $\varphi$ is unsatisfiable (unsat) or if its $\delta$-weakening $\varphi^\delta$ is satisfiable ($\delta$-sat) (Gao et al. (2012)). Here, the $\delta$-weakening of $\varphi$ is defined as

$$\varphi^\delta := (\exists z \in Z) \left( \bigwedge\nolimits_{i=1}^{m} \left( \bigvee\nolimits_{j=1}^{k_i} |g_{ij}(x)| \leq \delta_{\text{SMT}} \right) \right), \tag{20}$$

where $\delta$ is a positive rational number. The cases of unsat and $\delta$-sat can overlap, in which case dReal can return either answer. Since we aim to find unsat, this overlap can result in 'false negatives'. The overlap occurs for tight inequalities, i.e. $\varphi' = (\forall x \in C)f(x) \leq 0$ is satisfied and there exists a $x \in C$ such that $f(x) = 0$. We will briefly demonstrate this overlap. Taking the negation of $\varphi'$ and rewriting it in the form (19) yields:

$$(\exists x \in C)f(x) > 0 \equiv (\exists x \in C, \exists z_1 \in Z_1)f(x) - z_1 = 0,$$

where $Z_1 = (0, m_1]$, $m_1 > \sup_{x \in C} f(x)$. Now since $\sup_{x \in C} f(x) = 0$, it is obvious that for any rational number $\delta_{\text{SMT}}$ we satisfy $(\exists x \in C, \exists z_1 \in Z_1)|f(x) - z_1| \leq \delta_{\text{SMT}}$. Hence, $\neg\varphi'$ is both unsat and $\delta$-sat. To circumvent this overlap, we introduced the $\delta$ relaxation in (3) to relax potential tight inequality bounds. Here, we pick $\delta > \delta_{\text{SMT}}$. To demonstrate the effect, consider again the example of $f(x)$, but now we want to verify $(\exists x \in C)f(x) - \delta \leq 0$. Hence, we have:

$$\begin{aligned}
\varphi &= (\exists x \in C, \exists z_2 \in Z_2)f(x) - \delta - z_2 = 0 \\
\varphi^\delta &= (\exists x \in C, \exists z_2 \in Z_2)|f(x) - \delta - z_2| \leq \delta_{\text{SMT}}
\end{aligned}$$

where $Z_2 = (0, m_2]$, $m_2 > \sup_{x \in C} f(x) - \delta = -\delta$. Since $f(x) \leq 0$, we picked $\delta > \delta_{\text{SMT}}$, and either $Z_2 = \emptyset$ or $z_2 > 0$, it is easy to see we have unsat but don't have $\delta$-sat. Therefore there is no overlap.

In the case of $\delta$-sat, dReal also returns a domain on which $\varphi^\delta$ is satisfied. Elements from this set can be used as counterexamples where the inequalities are potentially violated.

### 5.2 Fitness

The fitness value of an individual is given by

$$f_{\text{full}}(k) = \sum_{i=1}^{6} w_i(k) f_{\text{samp},\phi_i}(k) + f_{\text{smt},\phi_i}(k), \tag{21}$$

where $k$ is the current generation, $f_{\text{samp},\phi_i}(k) \in [0,1]$ a fitness based on the error measure $e_{\phi_i}(x_{t,i})$ defined in (18), $f_{\text{smt}} \in \{0,1\}$ a fitness based on the answer returned by the SMT solver dReal, and $w_i(k)$ gives weight to inequalities that were harder to satisfy in previous runs.

The fitness $f_{\text{samp},\phi_i}(k)$ is defined as

$$f_{\text{samp},\phi_i}(k) = \frac{1}{1+e_{\phi_i}(x_{t,i})}. \tag{22}$$

Here $x_{t,i}$ consists of randomly sampled points in $C_i$ and the counterexample domains returned by the SMT solver. The fitness $f_{\text{smt},\phi_\text{i}}(k)$ is defined to be 1 if the negation of (16) is unsatisfied and 0 otherwise. Let us denote the fitness of the $j$th individual as $f^j_{\text{samp},\phi_i}$. The $i$th fitness error of the best individual is defined as $e_{f,i}(k) = 1 - f^{j^*}_{\text{samp},\phi_i}(k)$, where $j^* = \arg\max_j \sum_{i=1}^{6} f^j_{\text{samp},\phi_i}(k)$. To give weight to inequalities that are farther away from satisfiability, we define a weight $\Delta_i(k)$ proportional to the contribution of $e_{f,i}(k)$ to the sum of all fitness errors:

$$\Delta_i(k) = \begin{cases} 6\dfrac{e_{f,i}(k)}{\sum_{i=1}^{6} e_{f,i}(k)} & \text{if } \displaystyle\sum_{i=1}^{6} e_{f,i}(k) \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{23}$$

To prevent strong fluctuation of the weights, we apply a discrete low-pass filter, yielding the following dynamic weight $w_i(k)$:

$$w_i(k) = \lambda\Delta_i + (1-\lambda)w_i(k-1), \tag{24}$$

where $w(0) = 1$ and a user-defined $\lambda \in [0,1]$. By increasing the weight of criteria that under-performed in previous generations, the selection probability of individuals that perform better on these criteria increases.

### 5.3 *Automatic synthesis*

Given a system, sets $S, I, G$, and $\delta$, the proposed approach consists of the following steps:

(1) A random population of individuals is constructed as described in Section 4. Each individual has one gene for each of the following: the CLF, the controller modes, and the parameter vector $[\alpha, \beta, \gamma, \zeta, \varepsilon]$.
(2) The fitness (21) is evaluated for all individuals.
(3) The SMT solver is used to generate counterexamples for the individual with the best fitness and all individuals with $\sum_{i=1}^{6} f_{\text{samp},\phi_i} = 6$.
(4) Elitism: the best individual is copied to the next generation.
(5) Based on the fitness, individuals are selected using tournament selection (Koza (1992)) to form new individuals by means of genetic operators. This step is repeated until a new full population is created.
(6) Steps 2 to 5 are repeated until all inequalities in (3) are satisfied or a maximum amount of generations is met. In the latter case, no guarantees are provided.

## 6. EXAMPLE: 2D PENDULUM

In this section we demonstrate our approach on a simple 2D pendulum, which demonstrates the ability to handle non-polynomial functions. The equations of motion of the pendulum system are given by:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= (-bx_2 + mlg\sin(x_1) + u)/J \end{aligned} \tag{25}$$

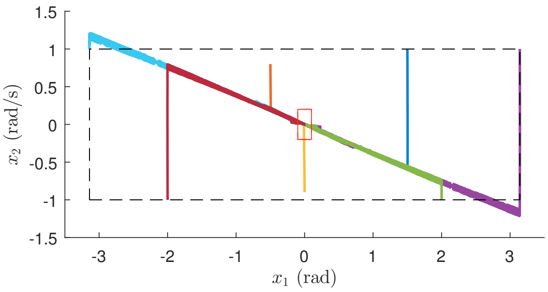

Fig. 3. Phase plot of multiple initial conditions. Black dashed: initial set, red: goal set.

where $u$ is the input and $x_1 = 0$ corresponds to upper position of the pendulum. The parameters of the system are chosen to be $m = 1$kg, $l = 1$m, $J = 1\text{kg}\cdot\text{m}^2$, $g = 9.81\text{m}^2/\text{s}$, and $b = 0.1\text{kg}\cdot\text{m}^2/\text{s}$.

The safe set, initial set, and goal set are given by $S = [-1.5\pi,\ 1.5\pi] \times [-10,\ 10]$, $I = [-\pi,\ \pi] \times [-1,\ 1]$, and $G = [-0.1,\ 0.1] \times [-0.2,\ 0.2]$ respectively. The design parameters are chosen to be $\delta = 0.0001$, $\delta_{\text{SMT}} = 0.00001$, and $\lambda = 0.05$. We fixed $\zeta = \delta$. The test points in the vector $x_{t,i}$ consist of 2000 uniformly sampled points from $C_i$ and an additional maximum 2000 counterexamples obtained from dReal, where a first-in-first-out principle is used. The number of individuals is 100 and maximum amount of generations is set to 2000. The mutation and crossover chances are picked to be 0.7 and 0.2 respectively. The used production rules are shown in Table 2, where the start symbols for the CLF, controller modes, and parameter vector $[\alpha, \beta, \gamma, \varepsilon]$ are given by $S_V = \langle\text{Vexpr}\rangle$, $S_u = \langle\text{mode}\rangle$, and $S_{\text{par}} = \langle\text{parvec}\rangle$ respectively. The terminal production rules are obtained from the production rules by omitting all the recursive rules. The GGGP algorithm is implemented in Mathematica. Running on an Intel Xeon processor with 6 cores and 3.5GHz, the statistics of 50 runs are shown in Table 3.

An example of a found solution is

$$\begin{aligned} V =& 38x_1^2 - 0.057x_1x_2^3 + 6.8x_1x_2 \\ & - 0.0041x_1x_2\sin(x_1) + 8.8x_2^2, \\ H =& \{6.9x_2,\ 82x_1,\ \ -89x_1,\ \ -95x_2\}, \\ \alpha =& 0.068,\ \ \beta = 0.064,\ \ \gamma = 690,\ \ \varepsilon = 7.2\cdot 10^{-12}. \end{aligned}$$

A phase plot of the closed-loop system for the initial conditions $(1.5, 1)$, $(-0.5, 0.8)$, $(0, -0.9)$, $(2, -1)$, $(-2, -1)$, $(\pi, 1)$, and $(-\pi, 1)$ is shown in Figure 3. It can be seen that all trajectories reach and stay in the goal set $G$.

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

| Nonterminal | Production rules |
|---|---|
| $\langle\text{Vexpr}\rangle ::=$ | $\langle\text{Vexpr}\rangle + \langle\text{Vexpr}\rangle \mid \langle\text{pol}\rangle$ |
| $\langle\text{mode}\rangle ::=$ | $\{\langle\text{lin}\rangle\} \mid \{\langle\text{lin}\rangle, \langle\text{lin}\rangle\} \mid \ldots$ |
| $\langle\text{parvec}\rangle ::=$ | $[\langle\text{par}\rangle, \langle\text{par}\rangle, \langle\text{par}\rangle, \langle\text{par}\rangle]$ |
| $\langle\text{pol}\rangle ::=$ | $\langle\text{pol}\rangle + \langle\text{pol}\rangle \mid \langle\text{const}\rangle \times \langle\text{mon}\rangle$ |
| $\langle\text{lin}\rangle ::=$ | $\langle\text{const}\rangle\, x_1 + \langle\text{const}\rangle\, x_2 \mid \langle\text{const}\rangle\, x_1$ |
| | $\mid \langle\text{const}\rangle\, x_2 \mid \langle\text{const}\rangle + \langle\text{lin}\rangle$ |
| $\langle\text{mon}\rangle ::=$ | $\langle\text{var}\rangle \mid \langle\text{var}\rangle \times \langle\text{mon}\rangle \mid x_1^2 \mid x_2^2 \mid x_1x_2$ |
| $\langle\text{var}\rangle ::=$ | $x_1 \mid x_2 \mid \sin x_1$ |
| $\langle\text{const}\rangle ::=$ | $0.1\,\langle\text{const}\rangle \mid 10\,\langle\text{const}\rangle \mid \langle\text{sign}\rangle \times (\langle\text{digit}\rangle\,\langle\text{digit}\rangle)$ |
| $\langle\text{par}\rangle ::=$ | $0.1\,\langle\text{par}\rangle \mid 10\,\langle\text{par}\rangle \mid (\langle\text{digit}\rangle\,\langle\text{digit}\rangle)$ |
| $\langle\text{sign}\rangle ::=$ | $-1 \mid 1$ |
| $\langle\text{digit}\rangle ::=$ | $0 \mid 1 \mid \ldots \mid 9$ |

Table 3. Results for 50 runs.

**Legend:** $t_T$: total run time, $\bar{t}_{\text{gen}}$: average generation time, # Gen: number of generations, $t_d$: minimum dwell-time, $\mu$: mean, $\sigma$: standard deviation.

| | Min | Max | $\mu$ | $\sigma$ |
|---|---|---|---|---|
| $t_T$ [min] | 3.97 | 143.61 | 31.46 | 32.60 |
| $\bar{t}_{\text{gen}}$[s] | 7.83 | 16.08 | 10.23 | 1.68 |
| # Gen | 22 | 756 | 177 | 167 |
| $\tau_d$ [s] | $3.1 \cdot 10^{-26}$ | $1.1 \cdot 10^{-9}$ | $2.2 \cdot 10^{-11}$ | $1.5 \cdot 10^{-10}$ |

## 7. DISCUSSION AND CONCLUSION

In this paper we presented an automatic synthesis approach of both a CLF and switched state feedback controller for nonlinear systems to satisfy both RSWS properties and no Zeno behavior. Preliminary results have been shown for a 2D pendulum.

Relying on genetic programming, the proposed methodology has as a drawback that there is no guarantee a solution is found within an arbitrary large number of generations.

In future work, we want to test the approach for multiple systems and investigate the scalability for higher dimensional systems. In the current framework, parameters and constants are synthesized using a grammar. We would like to allow for a separate optimization of parameters to find better solutions faster. Furthermore, the current framework imposes a conservative bound on $\ddot{V}_q(x)$, yielding a very low bound on the minimum-dwell time. Less conservative bounds, the extension to more general LTL formulae, and extensions to classes of hybrid systems will be explored in the future.

## REFERENCES

Ahmadi, A.A., Krstic, M., and Parrilo, P.A. (2011). A globally asymptotically stable polynomial vector field with no polynomial lyapunov function. In *Proc. of the IEEE Conference on Decision and Control*, 7579–7580.

Artstein, Z. (1983). Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11), 1163 – 1173.

Barrett, C.W., Sebastiani, R., Seshia, S.A., and Tinelli, C. (2009). Satisfiability modulo theories. *Handbook of satisfiability*, 185, 825–885.

Chen, P. and Lu, Y.Z. (2011). Automatic design of robust optimal controller for interval plants using genetic programming and kharitonov theorem. *International Journal of Computational Intelligence Systems*, 4(5), 826–836.

Chesi, G. (2009). Estimating the domain of attraction for non-polynomial systems via LMI optimizations. *Automatica*, 45(6), 1536 – 1541.

Diveev, A.I. and Shmalko, E.Y. (2015). Automatic synthesis of control for multi-agent systems with dynamic constraints. *IFAC-PapersOnLine*, 48(11), 384–389.

Gao, S., Avigad, J., and Clarke, E.M. (2012). $\delta$-complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, 286–300. Springer.

Gao, S., Kong, S., and Clarke, E.M. (2013). dreal: An smt solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*, 208–214. Springer.

Grosman, B. and Lewin, D.R. (2009). Lyapunov-based stability analysis automated by genetic programming. *Automatica*, 45(1), 252 – 256.

Hancock, E.J. and Papachristodoulou, A. (2013). Generalised absolute stability and sum of squares. *Automatica*, 49(4), 960 – 967.

Khalil, H. (2002). *Nonlinear Systems*. Pearson Education. Prentice Hall.

Koza, J.R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., and Lanza, G. (2003). *Genetic programming IV: Routine human-competitive machine intelligence*, volume 5. Springer US.

Mazo Jr, M., Davitian, A., and Tabuada, P. (2010). Pessoa: A tool for embedded controller synthesis. In *International Conference on Computer Aided Verification*, 566–569. Springer.

Mouelhi, S., Girard, A., and Gössler, G. (2013). Cosyma: A tool for controller synthesis using multi-scale abstractions. In *Proc. of the 16th International Conference on Hybrid Systems: Computation and Control*, 83–88. ACM.

Naur, P. (1963). Revised report on the algorithm language algol 60. *Commun. ACM*, 6(1), 1–17.

Papachristodoulou, A. and Prajna, S. (2002). On the construction of lyapunov functions using the sum of squares decomposition. In *Proc. of the IEEE Conference on Decision and Control*, volume 3, 3482–3487 vol.3.

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

Romdlony, M.Z. and Jayawardhana, B. (2016). Stabilization with guaranteed safety using control lyapunovbarrier function. *Automatica*, 66, 39 – 47.

Rungger, M. and Zamani, M. (2016). Scots: A tool for the synthesis of symbolic controllers. In *Proc. of the 19th International Conference on Hybrid Systems: Computation and Control*, 99–104. ACM.

Sekaj, I. and Perkacz., J. (2007). Genetic programming - based controller design. In *2007 IEEE Congress on Evolutionary Computation*, 1339–1343.

Tabuada, P. (2009). *Verification and control of hybrid systems: a symbolic approach*. Springer US.

Whigham, P.A. et al. (1995). Grammatically-based genetic programming. In *Proc. of the workshop on genetic programming: from theory to real-world applications*, 33–41.

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

Xu, X., Tabuada, P., Grizzle, J.W., and Ames, A.D. (2015). Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27), 54 – 61.