Robust Controller Synthesis of Switched Systems Using Counterexample Guided Framework

Hadi Ravanbakhsh and Sriram Sankaranarayanan
University of Colorado, Boulder
firsname.lastname@colorado.edu

ABSTRACT

We investigate the problem of synthesizing robust controllers that ensure that the closed loop satisfies an input reach-while-stay specification, wherein all trajectories starting from some initial set I, eventually reach a specified goal set G, while staying inside a safe set S. Our plant model consists of a continuous-time switched system controlled by an external switching signal and plant disturbance inputs. The controller uses a state feedback law to control the switching signal in order to ensure that the desired correctness properties hold, regardless of the disturbance actions.

Our approach uses a proof certificate in the form of a robust control Lyapunov-like function (RCLF) whose existence guarantees the reach-while-stay specification. A counterexample guided inductive synthesis (CEGIS) framework is used to find a RCLF by solving a $\exists\forall\exists\forall$ formula iteratively using quantifier free SMT solvers. We compare our synthesis scheme against a common approach that fixes disturbances to nominal values and synthesizes the controller, ignoring the disturbance. We demonstrate that the latter approach fails to yield a robust controller over some benchmark examples, whereas our approach does.

Finally, we consider the problem of translating the RCLF synthesized by our approach into a control implementation. We outline the series of offline and real-time computation steps needed. The synthesized controller is implemented and simulated using the Matlab(tm)/Simulink(tm) model-based design framework, and illustrated on some examples.

1. INTRODUCTION

The problem of *correct-by-construction* controller design seeks a feedback control law that controls a given plant model to satisfy a property specification. In this paper, we examine the problem of designing controllers that robustly control a switched system, which is subject to external disturbance inputs, lying inside a set D and a controlled switching mode from a finite set Q. The property specification is a *reach-while-stay* (RWS) property that states that all traces of the resulting closed loop starting from a set I will remain inside a safe set S until, eventually reach a goal set G.

The approach is to use a robust control Lyapunov-like function (RCLF) $V(\mathbf{x})$ defined over the plant states. For any state, we require that there be a switching mode q which can be chosen to strictly decrease the value of V, as long as the controller remains in the set $S \setminus G$. Additionally, we require that the value of V be negative inside the initial set I and positive at the boundary of the safe set S. These conditions together naturally guarantee the existence of a controller that can switch between appropriate control modes to satisfy the RWS property. Furthermore, we prove that such a controller will naturally satisfy a minimum dwell time property under the assumption that the sets I, S and G are all compact.

Next, we present an approach that will automatically discover an RCLF. This is posed as a constraint feasibility problem wherein the form of the function V is specified as a polynomial with unknown coefficients $\mathbf{c}$. However, we require the solution to a set of nonlinear quantified constraints. It is well known that these constraints are hard to solve even for small systems. We provide an extension of a now well-known idea called counter-example guided inductive synthesis (CEGIS) [25, 26]. CEGIS was originally proposed as a program synthesis technique that can solve $\exists\forall$ constraints to synthesize values for missing constants in *program sketches* so that the resulting program satisfies a set of assertions. Recently, we showed the applicability of the CEGIS approach to control synthesis, but in the absence of disturbance inputs [22]. In this paper, we extend the approach to solve problems with disturbances. In theory, the presence of disturbances requires us to consider an extra quantifier alternation yielding $\exists\forall\exists\forall$ constraints. As a result, it is computationally challenging to naively extend CEGIS approach designed for $\exists\forall$ constraints. We show that an extension that modifies the structure of the witnesses in the CEGIS procedure, can be applied to naturally handle this quantifier elimination.

We implement our approach using a combination of SMT solver Z3 [7] and an off-the-shelf linear matrix inequality (LMI) solver to synthesize RCLFs for benchmark examples. We compare our work with a default approach that ignores disturbances by setting them to a nominal fixed value. We show that the certificates thus obtained are not robust to disturbances in many situations. In contrast, our approach succeeds in finding a robust CLF.

Finally, we turn our attention to the problem of synthesizing the controller implementation from the RCLF. This problem is not considered in [22], and is shown to be nontrivial, especially when the worst case execution times of the various components need to be taken into account to guarantee that the mode switches happen at the appropriate time instants. We design a time triggered implementation and illustrate it on two benchmark examples.

1.1 Related Work

Fundamentally, there are two types of approaches to solving control synthesis problems. The first is based on a combination of abstraction to a finite game graph and solving this finite game using

EMSOFT'16, October 01-07, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2968478.2968485

fixed-point algorithms. One advantage of this approach is the ability to easily handle complex temporal specifications [16]. In this technique, the system is usually abstracted using a simulation relation. The abstraction guarantees that a solution (switching strategy) for the abstract system is also a solution for the original system. In the next step, a solution for the abstract system is calculated using fixed-point computation. Most of these techniques do not consider disturbances because of computational difficulties in solving two player games [17, 3, 21]. However, the uncertainties can be easily modeled in the abstraction and this, makes it easy to consider disturbances as well [30, 16].

Another methodology is based on solving constraints. In the first phase, the problem is reduced to finding a control certificate (e.g. control Lyapunov function). A control certificate provides a control strategy and a certificate, which guarantees the specification. Dimitrova and Majumdhar provide a proof system for ATL* properties using a combination of control Lyapunov-like functions [8]. However, for the most part it has proven hard to manually generate such certificates. As a result, the approach in this paper, focuses on a smaller class of constraints while providing automated approaches for the controller synthesis problem. As mentioned earlier, we directly extend our previous work [22] in two ways: (a) we handle disturbance inputs and the attendant quantifier alternation involved, and (b) we provide a solution to the control implementation synthesis from the certificates.

In this paper, we use a control Lyapunov-like function (CLF) to guarantee the specification. The idea of using CLFs was originally proposed by Artstein [2] and Sontag [27] showed how to design a controller from a given CLF. Later, the concept of *robust* CLFs (RCLF) was introduced [24, 9]. Battilotti [4, 5] showed how one can design robust controls using RCLFs. In most of these methods, the control input has a fixed form. This fixed form can be a polynomial (in state variables) which yields a static feedback controller[29, 12] or a control table [14].

The approach of Taly et al. synthesizes the missing parts of a skeleton switching logic that allows the overall system to satisfy some control objectives [28]. The similarities with our method include the use of constraint solvers to encode the presence of Lyapunov and barrier functions to guarantee reachability and safety properties. In this work, no such skeleton is specified and furthermore, we are able to deal with disturbances.

As mentioned earlier, this work extends the so-called counterexample guided inductive synthesis (CEGIS) procedure. The CEGIS procedure has been primarily used for parameter synthesis in programming languages (e.g. [25, 1]) and hybrid systems [10, 31]. More recently, a similar procedure used for finding Lyapunov functions by Kapinski et. al [15], and control Lyapunov functions [22] solving $\exists\forall$ formulae. In this work we use CEGIS procedure to solve $\exists\forall\exists\forall$ formulae when searching for RCLFs. We compare our approach to the "default" strategy that synthesizes a controller by nominally fixing a disturbance value and verifying the robustness of the resulting controller.

2. BACKGROUND

In this section, we discuss the model used for the switched system along with reach-while-stay objectives. We then discuss proof rules for enforcing the specification in the absence of disturbances.

We use $\mathbb{R}$ to denote the real numbers. For a set $X \subseteq \mathbb{R}^n$, we denote its interior as $int(X)$. For a function $f(t)$, let $f^+(t)$ refer to the right limit $\lim_{s\to t^+} f(s)$. Let $\dot{f}(t)$ be the right derivative of f wrt t, i.e, $\lim_{s\to t^+} \frac{f(s)-f(t)}{s-t}$.

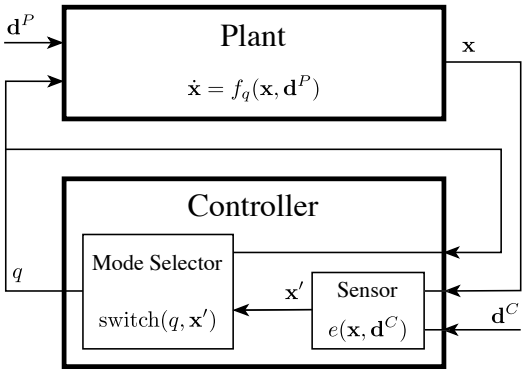


Figure 1: State-feedback switched system

2.1 System Model

The system model is a switched system consisting of a plant and a controller. The plant has n continuous state variables written as $\mathbf{x} \in X \subseteq \mathbb{R}^n$. The inputs to the plant are (i) the (controllable) mode of the plant which belongs to finite set Q and (ii) the (uncontrollable) plant disturbance $\mathbf{d}^P \in D^P \subset \mathbb{R}^n$. The state of the plant updates according to the dynamics for the current mode of the system.

DEFINITION 1 (PLANT). *A plant $\Psi(X, Q, D^P, f)$ is defined over a state space X, finite set of modes Q, compact disturbance space D^P ($\mathbf{0} \in D^P \subset \mathbb{R}^n$) and (iv) a map from inputs and state to vector field $f : X \times Q \times D^P \to \mathbb{R}^n$.*

For simplicity, we write $f_q(\mathbf{x}, \mathbf{d}^P)$ instead of $f(\mathbf{x}, q, \mathbf{d}^P)$. Also, we assume that $f_q(\mathbf{x}, \mathbf{d}^P)$ is a polynomial in $\mathbf{x}$ and $\mathbf{d}^P$.

The plant disturbance input signal is a function $\mathbf{d}^P(.) : \mathbb{R}^+ \to D$, mapping time t to the disturbance input $\mathbf{d}^P(t) \in D$. The control input signal $q(.) : \mathbb{R}^+ \to Q$ is a piecewise constant function which maps time t to the control mode $q(t)$. The state $\mathbf{x}(.) : \mathbb{R}^+ \to X$ is a function that satisfies

$$\dot{\mathbf{x}}(t) = f_{q(t)}(\mathbf{x}(t), \mathbf{d}(t)) \,.$$

The controller is memoryless, and has two parts: (i) the sensor which measures (estimates) the state of the plant $\mathbf{x}$ and (ii) the mode selector. The sensor has inputs (a) the plant state $\mathbf{x}$ and (b) the (uncontrollable) control disturbance $\mathbf{d}^C$, modeling the measurement error. The sensor then provides $\mathbf{x}'$ (measured state) as output. The mode selector receives the measured state $\mathbf{x}'$ and the current mode. The mode for the next time instant is its output. A schematic view of the closed loop system is shown in Figure 1.

DEFINITION 2 (CONTROLLER). *The controller is defined over a state space X, set of modes Q, compact control disturbance space D^C ($\mathbf{0} \in D^C \subset \mathbb{R}^n$), sensor function $e : X \times D^C \to X$ which maps plant state and control disturbance to measured plant state, and mode selector function* switch $: Q \times X \to Q$ *which decides the new control mode for the system, given measured state and current mode. Formally, let $\kappa(X, Q, D^C, e,$* switch*$)$ denote a controller.*

Similar to plant disturbance, control disturbance is uncontrollable and $\mathbf{d}^C(.) : \mathbb{R}^+ \to D^C$ is any function describing the control disturbance for all times.

REMARK 1. *It is feasible to extend the control model to consider (a) state estimation error wherein the state is estimated from*

an output through a filter, (b) delays in computing the control mode from a given state and (c) delays in a commanded state from taking effect. For instance, (a) and (b) are handled using an appropriate control disturbance and (c) through a plant disturbance input.

In a short form, the memoryless controller is a function and trace of the mode satisfies condition below

$$q^+(t) = \mathsf{switch}(q(t), e(\mathbf{x}(t), \mathbf{d}^C(t)))\,.$$

Notice that given $\mathbf{x}(0)$, $q(0)$, $\mathbf{d}^P(.)$ and $\mathbf{d}^C(.)$, the traces of the system is unique. The control synthesis problem is to design function switch s.t. the closed loop system satisfies the specification.

2.2 Control Synthesis Problem

A specification φ is described over the trace of state $\mathbf{x}(.)$. The specification in this article is reach-while-stay w.r.t. compact regions $S(\subseteq X)$, $I(\subseteq int(S))$, $G(\subseteq I)$ as safe, initial and goal sets, respectively.

DEFINITION 3 (REACH-WHILE-STAY). *$RWS(S, I, G, \mathbf{x}(.))$ is a specification that guarantees a trace, starting from region I, reaches G, while staying in S. $RWS(S, I, G, \mathbf{x}(.))$ is defined as*

$$\mathbf{x}(0) \in I \implies (\exists T) \begin{pmatrix} \mathbf{x}(T) \in G \\ (\forall 0 \leq t < T)\, \mathbf{x}(t) \in S \end{pmatrix}.$$

DEFINITION 4 (SYNTHESIS PROBLEM). *The problem of controller synthesis is to find* switch *function such that a specification $RWS(S, I, G, \mathbf{x}(.))$ is guaranteed: Find* switch *s.t.*

$$(\forall \mathbf{x}(0), q(0), \mathbf{d}^P(.), \mathbf{d}^C(.))\; RWS(S, I, G, \mathbf{x}(.))\,. \qquad (1)$$

Figure 2 shows different regions for reach-while-stay along with some traces which respect the specification.

2.3 Proof Rules

The specification $RWS(S, I, G, \mathbf{x}(.))$ combines liveness and safety aspects that require the trace to eventually reach G, while staying within the set S. Proof rules involving Lyapunov-like functions for RWS properties are well-known [8, 22]. We recall a proof rule that does not involve disturbance inputs:

DEFINITION 5 (LYAPUNOV-LIKE FUNCTION). *A Lyapunov-like function (LF) for $RWS(S, I, G, \mathbf{x}(.))$ is a polynomial function V that*

$$\begin{cases} \mathbf{x} \in I \implies V(\mathbf{x}) < 0 \\ \mathbf{x} \in \partial S \implies V(\mathbf{x}) > 0 \\ \mathbf{x} \in S \setminus G \implies \nabla V.f(\mathbf{x}) < -\epsilon\,. \end{cases} \qquad (2)$$

A Lyapunov-like function guarantees that $RWS(S, I, G, \mathbf{x}(.))$ holds in absence of inputs. It enforces that the value of the function is negative initially and is positive whenever the system reaches the boundary of S. Furthermore, the derivative of V is negative inside $S \setminus G$. Together, these premises guarantee that (a) G must be eventually reached and (b) the boundary of S will never be reached by a trace starting from I. In fact, the set $\{\mathbf{x} \mid V(\mathbf{x}) = 0\} \cap S$ forms a barrier that ensures that the boundary of S is never reached. The dashed line in Fig. 2 defines places in S s.t. $V(\mathbf{x}) = 0$ and the value of V decreases as time passes.

We will now define a Lyapunov-like function V in presence of control inputs. The first two constraints in Eq. (2) will be the same when the plant has a control input q.

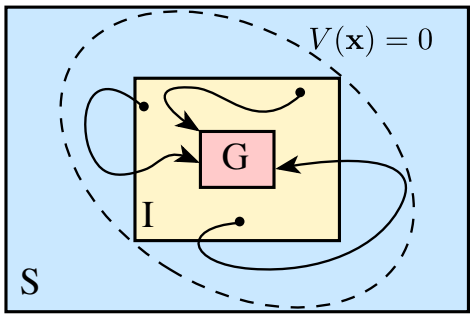


Figure 2: Reach-while-stay Specification w.r.t safe set S, initial set I and goal set G.

DEFINITION 6 (CONTROL LYAPUNOV-LIKE FUNCTION). *A control Lyapunov-like function (CLF) for $RWS(S, I, G, \mathbf{x}(.))$ is a polynomial function V that*

$$\begin{cases} \mathbf{x} \in I \implies V(\mathbf{x}) < 0 \\ \mathbf{x} \in \partial S \implies V(\mathbf{x}) > 0 \\ \mathbf{x} \in S \setminus G \implies (\exists q \in Q)\; \nabla V.f_q(\mathbf{x}) < -\epsilon\,. \end{cases} \qquad (3)$$

Comparing with Def. 5, we notice that the only change occurs in the last rule. Rather than requiring that $\nabla V.f$ is negative, we now require that at each state $\mathbf{x}$, a mode q is available to enforce decrease of V ($\nabla V.f_q$ is negative).

However, Defs. 5 and 6 both do not consider disturbance inputs. The presence of disturbances (plant disturbance and control disturbance) adds one more level of difficulty to the problem, involving a quantifier alternation. We now present strategies to synthesize in the presence of disturbances.

3. HANDLING DISTURBANCES

In this section, we discuss the handling of disturbances for robust implementation of controllers. For simplicity, let $\mathbf{d}$ be a vector describing the joint plant disturbance $\mathbf{d}^P$ and control disturbance $\mathbf{d}^C$. I.e, $\mathbf{d} \in D :\; D^P \times D^C$. Also, we use $f_q^D(\mathbf{x}, \mathbf{d})$ to represent $f_q(e(\mathbf{x}, \mathbf{d}^C), \mathbf{d}^P)$, assuming e is a polynomial.

The idea is to find a control mode q such that the value of V decreases under all possible values of disturbances. We modify Def. 6 to incorporate disturbances for the property $RWS(S, I, G, \mathbf{x}(.))$.

DEFINITION 7 (*Robust* CLF).*A robust CLF (RCLF) is a polynomial function V with the following properties:*

$$\begin{cases} \mathbf{x} \in I \implies V(\mathbf{x}) < 0 \\ \mathbf{x} \in \partial S \implies V(\mathbf{x}) > 0 \\ \mathbf{x} \in S \setminus G \implies \\ \qquad (\exists q \in Q)\; (\forall \mathbf{d} \in D)(\nabla V).f_q^D(\mathbf{x}, \mathbf{d}) < -\epsilon\,, \end{cases} \qquad (4)$$

When compared to Eq. (3), the third condition for the RCLF is more complicated because of extra $(\forall \mathbf{d})$ quantifier. We use the counter-example guided synthesis (CEGIS) framework in Section 4 to handle this quantifier alternation.

Let us define a function $\mathsf{cond}_q(\mathbf{x})$ over a state $\mathbf{x}$ and mode q as

$$\mathsf{cond}_q(\mathbf{x}) = \max_{\mathbf{d} \in D} (\nabla V).f_q^D(\mathbf{x}, \mathbf{d})\,. \qquad (5)$$

From Eq. (4), we note that the goal of a controller is to find a mode q that guarantees that $\mathsf{cond}_q(\mathbf{x}) < -\epsilon$. For a given mode q and state $\mathbf{x}$ the controller performs the following actions: (a) Evaluate

$\mathsf{cond}_q(\mathbf{x})$ and check if $\mathsf{cond}_q(\mathbf{x}) < -\epsilon_s$ for a *switching threshold value* ϵ_s ($0 < \epsilon_s < \epsilon$) supplied by the user. (b) If this fails, the controller finds a new mode $\hat{q} \in Q$ such that $\mathsf{cond}_{\hat{q}}(\mathbf{x}) < -\epsilon$. Otherwise, the mode remains q. The control function switch can be defined succinctly as

$$\mathsf{switch}(q, \mathbf{x}) := \begin{cases} \hat{q} & \text{if } \begin{pmatrix} \mathsf{cond}_q(\mathbf{x}) \geq -\epsilon_s \\ \wedge\, \mathsf{cond}_{\hat{q}}(\mathbf{x}) < -\epsilon \\ \wedge\, \mathbf{x} \in S \setminus G \end{pmatrix} \\ q & \text{otherwise}\,. \end{cases} \tag{6}$$

Now, we show that having a RCLF (V) Eq. (6) gives a controller that does not switch too fast. Assume that the sets S, G in the specification and disturbances D are all compact sets. Let V and f_q^D be bounded for each mode q. Let $\mathbf{x}(t)$ be a time trajectory of the closed loop system with corresponding switching function $q(t)$, using the switching law in Eq. (6).

LEMMA 1. *There exists a* minimum dwell time $\delta > 0$ *such that for any time T, if for all $t \in [T, T+\delta]$*
1. $\mathbf{x}(t) \in S \setminus G$
2. $q(t) = q(T)$,

and $\mathsf{cond}_{q(T)}(\mathbf{x}(T)) < -\epsilon$, *then*

$$(\forall t \in [T, T+\delta])\ \mathsf{cond}_{q(t)}(\mathbf{x}(t)) \leq -\epsilon_s)\,.$$

PROOF. Let $q = q(T)$. We are given that

$$\mathsf{cond}_{q(T)}(\mathbf{x}(T)) < -\epsilon\,,$$

and let $\tau > 0$ be any time such that

$$(\forall t \in [T, T+\tau])\ \mathsf{cond}_q(\mathbf{x}(t)) \leq -\epsilon_s\ \wedge\ q(t) = q\,.$$

Note that cond_q is a continuous and piecewise differentiable function of $\mathbf{x}$. As a result, there is a Lipschitz constant A_q such that

$$|\mathsf{cond}_q(\mathbf{x}(T+\tau)) - \mathsf{cond}_q(\mathbf{x}(T))| \leq A_q ||\mathbf{x}(T+\tau) - \mathbf{x}(T)||\,.$$

Now, $\mathbf{x}(t)$ in the interval $t \in [T, T+\tau]$ is the solution of an ODE $\frac{d\mathbf{x}}{dt} = f_q^D(\mathbf{x}, \mathbf{d}(t))$ and furthermore the state space is compact. As a result, there exists a constant B_q such that

$$\|\mathbf{x}(T+\tau) - \mathbf{x}(T)\| \leq B_q \tau\,.$$

Combining, we have $|\mathsf{cond}_q(\mathbf{x}(T+\tau)) - \mathsf{cond}_q(\mathbf{x}(T))| \leq \Lambda_q \tau$ wherein $\Lambda_q = A_q B_q$.

Let $\Lambda = \max_{q \in Q} \Lambda_q$. Let us choose a δ such that

$$\delta :\ \frac{\epsilon - \epsilon_s}{\Lambda}\,. \tag{7}$$

The above arguments show that for all $t \in [T, T+\delta]$,

$$|\mathsf{cond}_q(\mathbf{x}(T+t)) - \mathsf{cond}_q(\mathbf{x}(T))| \leq \Lambda_q t \leq \Lambda\delta \leq \epsilon - \epsilon_s\,.$$

Therefore, using that $\mathsf{cond}_q(\mathbf{x}(T)) < -\epsilon$, we obtain for all $t \in [T, T+\delta]$, $\mathsf{cond}_q(\mathbf{x}(T+t) \leq -\epsilon_s$. □

As a direct corollary, we can establish the following lemma.

LEMMA 2. *Let $0 \leq t_1 < t_2$ be any time interval such that the trajectory $\mathbf{x}(t) \in S \setminus G$ for $t \in [t_1, t_2]$.*
1. $(\forall t \in [t_1, t_2])\ \nabla V.f_{q(t)}^D(\mathbf{x}(t), \mathbf{d}(t)) \leq -\epsilon_s$.
2. *The controller respects min dwell time property.*

PROOF. Assume that $(\nabla V).f_{q(\tau)}^D(\mathbf{x}(\tau), \mathbf{d}(\tau)) > -\epsilon_s$ for some time $\tau \in [t_1, t_2]$. As a result,

$$\mathsf{cond}_{q(\tau)}(\mathbf{x}(\tau)) \geq (\nabla V).f_{q(\tau)}^D(\mathbf{x}(\tau), \mathbf{d}(\tau)) > -\epsilon_s\,.$$

Let $q(\tau) = q$. Let $\tau_0 < \tau$ be a switch time instant such that $\forall t \in (\tau_0, \tau], q(t) = q$. By Eq. (6), we note that $\mathsf{cond}_q(\mathbf{x}^+(\tau_0)) < -\epsilon < -\epsilon_s$. Because cond_q is a continuous function, there must be a time τ_1 such that $\tau_0 < \tau_1 < \tau$ and $\mathsf{cond}_q(\tau_1) = -\epsilon_s$. We note by assumption that $\mathbf{x}(\tau_1) \in S \setminus G$. As a result, by Eq. (6), we note that $q^+(\tau_1) \neq q(\tau_1)$, or in other words, a mode switch must happen at $\tau_1 < \tau$. This contradicts our assumption that $q(t) = q$ for all $t \in (\tau_0, \tau]$ and hence, $\nabla V.f_q^D(t)(\mathbf{x}(t), \mathbf{d}(t)) \leq -\epsilon_s$ for all $t \in [t_1, t_2]$.

By Lemma 1, there exists a $\delta > 0$ defined purely in terms of V, the dynamics at each mode and the set S such that whenever $\mathsf{cond}_q(\mathbf{x}(t)) < -\epsilon$, we have that $\mathsf{cond}_q(\mathbf{x}(t+\delta)) \leq -\epsilon_s$. As a result, we conclude that the time between two switches is at least δ. □

THEOREM 1. *Given compact sets G, I and S ($I \subseteq int(S)$) and a polynomial RCLF (V) satisfying Equation* (4)*, there is a control strategy guaranteeing min dwell time property s.t. $I \implies SUG$.*

PROOF. We show that there exists a set W s.t.
1. $I \implies WUG$.
2. $I \subseteq int(W)$.
3. $W \subseteq int(S)$.

W is defined as $W : (\{\mathbf{x} | V(\mathbf{x}) \leq 0\} \cap S$. By Equation (4), $W \subseteq int(S)$ and $I \subseteq int(W)$. Having Equation (4), according to Lemma 2, there exists a controller which respects the min dwell time property. Also, the controller guarantees $\nabla V.f_{q(t)}^D(\mathbf{x}(t), \mathbf{d}(t)) \leq -\epsilon_s$ as long as $\mathbf{x}(t) \in S \setminus G$. Therefore, if $\mathbf{x}(t) \in S \setminus G$ for $t \in [t_1, t_2]$ then $V(\mathbf{x}(t_2)) < V(\mathbf{x}(t_1)) - \epsilon_s(t_2 - t_1)$.

Now we show that $I \implies WUG$. Assuming otherwise for the sake of contradiction, we conclude that either the trace must remain in W forever or exit W without reaching G. Since $W \subseteq S$ is a compact set, the trace cannot remain in $W \setminus G$ forever. Otherwise, V must decrease forever but V is a continuous function that is lower bounded inside $W \setminus G$. Therefore, the trace must exit W without reaching G. By the continuity of the trajectory, it must reach ∂W. Let $T \geq 0$ be the first time instance s.t. $\mathbf{x}(T)$ reaches ∂W ($V(\mathbf{x}(T)) = 0$). Since $V(\mathbf{x}(0)) < 0$ ($\mathbf{x}(0) \in I$) and $(\forall t \in [0, T])\ \nabla V.f_{q(t)}^D(\mathbf{x}(t), \mathbf{d}(t)) < -\epsilon_s$, we have that $V(\mathbf{x}(T)) < V(\mathbf{x}(0)) - \epsilon_s T < V(\mathbf{x}(0)) < 0$. But if $\mathbf{x}(T) \in \partial W$, we have $V(\mathbf{x}(T)) = 0$, deriving a contradiction.

Therefore, state trace cannot stay forever in $W \setminus G$ and cannot exit W before entering G. While $\mathbf{x} \in (W \setminus G)$, by the construction of the controller, we can conclude time diverges (because the controller respects the min dwell time property). We conclude that the RWS property holds. □

4. COUNTEREXAMPLE GUIDED SYNTHESIS

So far, the problem of controller synthesis is reduced to the problem of finding RCLF V that satisfies Eq. (4). In this article, we restrict the search to polynomial RCLFs, which, in turn, leads to incompleteness of our method. Nevertheless, polynomial CLFs exists for many interesting problems [22]. Searching for a function V considers a parameterized function (*a template*) $V(\mathbf{c}, \mathbf{x})$: $\sum_\alpha \mathbf{c}_\alpha \mathbf{x}^\alpha$, where $\mathbf{c}_\alpha$ is a real-valued unknown coefficient and $\mathbf{x}^\alpha$ is a monomial in $\mathbf{x}$. The goal is to find parameters $\mathbf{c}^* \in C$ s.t $V(\mathbf{c}^*, \mathbf{x})$ satisfies Eq. (4). Formally, we seek to solve a quantified problem:

$$(\exists \mathbf{c} \in C)\ (\forall \mathbf{x} \in X)(\text{* Conditions from eq. (4) *})\,. \tag{8}$$

In this article we use a counterexample guided inductive synthesis (CEGIS) approach to solve our problem. The CEGIS approach

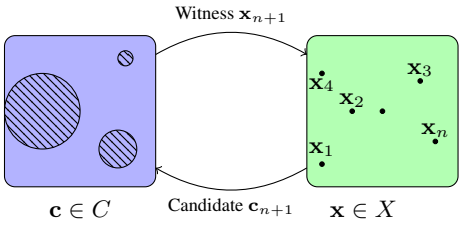


Figure 3: Illustration of the CEGIS iteration, showing the space of candidates, witnesses and the iterative synthesis process.

has been used widely in program synthesis for safety [25]. The key idea is to solve $\exists\forall$ formulae (such as in Eq. (8) above) by solving simpler unquantified problems, iteratively.

We now explain the general structure of the CEGIS framework for solving a constraint of the form

$$(\exists \mathbf{c} \in C)(\forall \mathbf{x} \in X)\ \Psi(\mathbf{c}, \mathbf{x})\,.$$

Figure 3 illustrates the iterative process. At each iteration, we maintain two sets: (a) A subset $C_n \subseteq C$ of the candidate space that is implicitly maintained as the set of solutions to an assertion $\psi_n[\mathbf{c}]$ and (b) A finite subset $X_n : \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of witnesses, such that $\mathbf{x}_i \in X$.

Initially, ψ_0 represents the entire candidate space C, and X_0 is the empty set. At each iteration, we perform three steps:

1. *Candidate Generation:* We generate a candidate point $\mathbf{c}_n \in C_n$ by solving the formula ψ_n and finding a satisfiable solution for it. Failure to obtain such a solution means that $C_n = \emptyset$ and we have thus run out of candidates.

2. *Witness Generation:* We check whether the candidate $\mathbf{c} = \mathbf{c}_n$ represents a valid candidate by checking:

   $$(\forall\, \mathbf{x} \in X)\Psi(\mathbf{c}_n, \mathbf{x})\,.$$

   More precisely, we check its negation $\neg\ \Psi(\mathbf{c}_n, \mathbf{x})$ as a formula involving only $\mathbf{x}$. If the negation is UNSAT, then we conclude that the current candidate $\mathbf{c}_n$ is a valid solution and stop. Otherwise, we add a witness $\mathbf{x}_{n+1}$.

3. *Refining Candidate Space:* Finally, we compute a new formula ψ_{n+1} that incorporates the witness as

   $$\psi_{n+1} : \bigwedge_{i=1}^{n+1} \Psi(\mathbf{c}, \mathbf{x}_i) = \psi_n \wedge \Psi(\mathbf{c}, \mathbf{x}_{n+1})\,.$$

   In other words, ψ_{n+1} forces future candidates $\mathbf{c}$ to be valid for all witnesses including $\mathbf{x}_{n+1}$.

For a more detailed discussion of CEGIS, its decidability and complexity please refer to [23, 22]. When the CEGIS terminates, it either gives a solution $\mathbf{c}$, or gives a set of witnesses X_n for which no candidate exists. As such, CEGIS approach cannot be used for the treatment of formula arising from Eq. (8), since it involves additional quantifier alternations of the form $(\exists \mathbf{c})\ (\forall \mathbf{x})\ (\exists\ q)\ (\forall \mathbf{d})$.

4.1 CEGIS for RCLF

We will now extend the original CEGIS framework to handle these further quantifier alternations. The idea is conceptually simple: we will extend the witnesses structure. Rather than witnesses which are simply points $\mathbf{x}_i \in X$, we will now allow witnesses that are of the form $(\mathbf{x}_i, (Q \mapsto D))$, i.e, a combination of a state $\mathbf{x}_i \in X$ and a map from each mode to a disturbance vector. Since Q is finite, this map is explicitly stored as $(\mathbf{x}_i, (q_1, \mathbf{d}_1), \ldots, (q_m, \mathbf{d}_m))$.

First, we note that the RCLF requirements yield formulae of the form

$$(\exists\ \mathbf{c})\ (\forall\ \mathbf{x})\Psi(\mathbf{c}, \mathbf{x})\,,$$

but Ψ itself is a quantified formula with the following structure:

$$\bigwedge \begin{cases} \mathbf{x} \in R_1 \implies \bigvee_q (\forall\ \mathbf{d})\ p_{1,q}(\mathbf{c}, \mathbf{x}, \mathbf{d}) < 0 \\ \mathbf{x} \in R_2 \implies \bigvee_q (\forall\ \mathbf{d})\ p_{2,q}(\mathbf{c}, \mathbf{x}, \mathbf{d}) < 0 \\ \vdots \\ \mathbf{x} \in R_r \implies \bigvee_q (\forall\ \mathbf{d})\ p_{r,q}(\mathbf{c}, \mathbf{x}, \mathbf{d}) < 0\,. \end{cases} \tag{9}$$

A first solution consists of applying CEGIS for $\exists\forall$ described previously. However, doing so yields quantified constraints for the candidate and witness generation steps. Since our objective was to avoid these quantified constraints in the first place, we modify the witness structure.

Witness Structure: As mentioned earlier, a witness to the violation of a given RCLF candidate $\mathbf{c} \in C$ include a state $\mathbf{x} \in X$ at which the violation happens along with for each mode q_j that can be selected for the mode, a disturbance witness $\mathbf{d}_j \in D$ that will violate the formula. With disturbances, each witness then has the following structure:

$$\mathbf{y}_i : \left(\mathbf{x}_i, (q_1, \mathbf{d}_1^{(i)}), \ldots, (q_m, \mathbf{d}_m^{(i)})\right)\,. \tag{10}$$

With this witness structure, the overall CEGIS procedure now extends naturally.

Candidate Generation: Let $\mathcal{Y}_n : \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of witnesses at the n^{th} iteration, starting from $\mathcal{Y}_0 : \emptyset$. The candidate is generated by solving the formula:

$$\psi_n[\mathbf{c}] :\ (\mathbf{c} \in C)\ \wedge \bigwedge_{i=1}^{n} \mathsf{inst}(\Psi, \mathbf{y}_i)\,, \tag{11}$$

wherein $\mathsf{inst}(\Psi, \mathbf{y}_i)$ substitutes the witness from Eq. (10) for the variables $\mathbf{x}$, q, $\mathbf{d}$ in Eq. (9).

$$\mathsf{inst}(\Psi, \mathbf{y}_i) : \begin{cases} \bigvee_{k=1}^{m} p_{1,q_k}(\mathbf{c}, \mathbf{x}_i, \mathbf{d}_k^{(i)}) < 0 \\ \bigvee_{k=1}^{m} p_{2,q_k}(\mathbf{c}, \mathbf{x}_i, \mathbf{d}_k^{(i)}) < 0 \\ \vdots \\ \bigvee_{k=1}^{m} p_{r,q_k}(\mathbf{c}, \mathbf{x}_i, \mathbf{d}_k^{(i)}) < 0\,. \end{cases} \tag{12}$$

We now use an SMT solver to find if the unquantified formula ψ_n from Eq. (11).

Witness Generation: Once a candidate $\mathbf{c} = \mathbf{c}_n$ is generated by solving ψ_n, we now evaluate if it is a true RCLF. This involves, substituting $\mathbf{c}_n$ for $\mathbf{c}$ in formula Ψ in Eq. (9) and checking if $\neg\Psi$ is satisfiable. Since Ψ itself is the conjunction of $r > 0$ conditions, its negation is a disjunction and we can check each disjunct separately for satisfiability. Each disjunct has the following form:

$$\mathbf{x} \in R_j\ \wedge \bigwedge_{k=1}^{m} (\exists\ \mathbf{d} \in D)\ p_{j,q_k}(\mathbf{c}_n, \mathbf{x}, \mathbf{d}) \geq 0\,.$$

We can remove the existential quantifier over $\mathbf{d}$ equivalently through $m = |Q|$ fresh set of variables $\mathbf{d}_1, \ldots, \mathbf{d}_m$. The new disjunct is written:

$$\mathbf{x} \in R_j\ \wedge \bigwedge_{k=1}^{m} p_{j,q_k}(\mathbf{c}_n, \mathbf{x}, \mathbf{d}_k) \geq 0\,. \tag{13}$$

If satisfiable, we obtain a witness $(\mathbf{x}_{n+1}, (q_1, \mathbf{d}_1^{(n+1)}), \ldots, (q_m, \mathbf{d}_m^{(n+1)}))$. Otherwise, we conclude that $\mathbf{c}_n$ represents a valid RCLF.

4.2 Solving Constraints

Finally, we consider the constraints that are solved during the process of CEGIS for generating RCLFs. We note that the template $V(\mathbf{c}, \mathbf{x})$ is a linear function over $\mathbf{c}$ but in general a polynomial over $\mathbf{x}$. As a result, when $\mathbf{x}$ and $\mathbf{d}$ are instantiated, each candidate generation problem ψ_n is a formula that involves Boolean combination of linear inequalities. Such a formula can be solved by efficient linear arithmetic SMT solvers such as Z3 [7].

The difficulty arises in evaluating whether the witness generation formula obtained by instantiating $\mathbf{c} = \mathbf{c}_n$ is satisfiable. This involves a conjunction of polynomial inequalities. As such SMT solvers such as dReal [11] and Z3 [7] can support the solution of these constraints. But the process is forbiddingly expensive, especially since it involves constraints over $|\mathbf{x}| + |Q||\mathbf{d}|$ variables.

In this regard, an idea previously proposed by authors can be used to extend this approach to larger systems [22]. This idea effectively introduces fresh variables corresponding to monomials $z_\alpha : \mathbf{x}^\alpha$. Each polynomial $p(\mathbf{x})$ is then written as a quadratic form $\mathbf{z}^t P \mathbf{z}$ where $\mathbf{z}$ collects the fresh variables corresponding to the monomials. Following this approach, we reduce Eq. (13) into a system of *linear matrix inequalities* (LMI) that can be solved efficiently using LMI solvers. Since LMIs are convex optimization problems, we can provide solutions to problems that have larger state and disturbance spaces.

4.3 Evaluation

We implemented the CEGIS framework using Z3 SMT solver [7] for candidate generation and Gloptipoly [13] as the LMI solver for finding witnesses. Gloptipoly is configured to use Mosek [18] as the SDP solver.

We compare the robust synthesis (RS) approach presented in this paper with a simple *Synthesize and Verify Robustness* (SVR) approach that uses a nominal disturbance value (e.g., $\mathbf{d} = 0$) and checks whether the resulting controller is robust, as a last step. Specifically, the disturbance free case uses Eq. (2) for synthesis. In doing so, we also check whether adding a "margin" by increasing the value of ϵ during controller synthesis necessarily makes the resulting design more robust to disturbances. For comparison, several examples are considered and all the experiments are carried out on a laptop with 2.9 GHz Intel Core i7 processor and 16GB of memory. The time limit is set to 5 hours.

The examples below use plant disturbances $D^P : [-r_D, r_D]^n$, with varying values of r_D. However, no control disturbances are added. The safe set S is $[-r_S, r_S]^n$. Likewise the goal set G and initial set I are spheres of radius r_G and r_I, respectively around specified center points. For all examples, we use template $V(\mathbf{c}, \mathbf{x}) = (\sum_{i \le j} c_{i,j}\ \mathbf{x}_i \mathbf{x}_j) - 1$.

EXAMPLE 1. *This problem instance is taken from [17] with two variables and a control input $u \in [-1, 1]$.*

$$\begin{bmatrix} \dot{\omega} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} \frac{B}{J} & \frac{k}{J} \\ -\frac{k}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u + \begin{bmatrix} d_\omega \\ d_i \end{bmatrix}$$

where $B = 10^{-4}$, $J = 25 \times 10^{-5}$, $k = 0.05$, $L = 15 \times 10^{-4}$ and $R = 0.5$. The desired operation point is $[\omega\ i] = [20\ \ 0]$ and region of interest is defined by $r_S : [10, 30] \times [-10, 10]$. By a change of basis, we set $(20, 0)$ as the new origin. In the new coordinate, S, G and I are defined by $r_S = 10$, $r_G = 0.5$ and $r_I = 4$, respectively with centers at the origin. There are two modes corresponding to $u \in \{-1, 1\}$. We choose $\epsilon = 0.01$.

Table 1: Results for Ex. 1 using RS method. Itr : # iterations, Time : total time (seconds).

r_D	Itr	Time	Status
0.0	3	36.1	✓
0.4	4	54.4	✓
0.8	5	115.3	✓
1.2	8	177.5	✓
1.4	34	800.9	✓
1.6	83	1500.0	✓
1.7	184	4367.8	✓
1.8	494	10565.4	✖

Table 2: Results for Ex. 1 using SVR method. Itr : # iterations, Time : total computation time (seconds).

ϵ	r_D	Itr	Time	Status
0.2	0.1	3	32.4	✓
0.3	0.2	5	40.7	✓
0.3	0.3	5	46.6	✖
0.4	0.0	536	5074.3	✖

The results of the RS method are shown in Table 1. To evaluate the effect of disturbances on the CEGIS procedure, we use different disturbance sizes. As results suggests, bigger disturbances impose harder restrictions on the RCLF and many more iterations are needed, as the size of disturbance gets bigger.

On the other hand, using the SVR technique, first a CLF is found with preferably higher values for ϵ. The most robust controller is obtained using $\epsilon = 0.03$ and it is verified that this controller can handle disturbances for $r_D = 0.2$. The results are shown in Table 2. These results suggest that RS method can provide provably robust controllers where the SVR approaches fails to synthesize controller for larger values of ϵ and fails to verify for larger disturbance values.

EXAMPLE 2. *We adopt this example from [20]. The plant consists of two variables x_1 and x_2 and three modes with the following dynamics*

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 + d_{x_1} \\ x_1 + d_{x_2} \end{bmatrix} + B_q$$

$$B_{q1} = \begin{bmatrix} 0 \\ -x_2^2 + 2 \end{bmatrix}, B_{q2} = \begin{bmatrix} 0 \\ -x_2 \end{bmatrix}, B_{q1} = \begin{bmatrix} 2 \\ 10 \end{bmatrix}.$$

The goal is to reach a region around $(-0.75, 1.75)$ ($G : \{(x_1, x_2) | (x_1 + 0.75)^2 + (x_2 - 1.75)^2 \le 0.25^2\}$). First, we change the bases and set $(-0.75, 1.75)$ as the new origin. Other parameters are $r_I = 1$ and $r_S = 2.25$.

For each method, we check for the biggest disturbance for which the problem can be solved. Using RS method, we were able to solve the problem when $r_D = 0.5$ using $\epsilon = 0.01$. For the SVR method, the most robust controller (obtained by setting $\epsilon = 0.2$) is verified to decrease V when $r_D = 0.03$. Detailed results are shown in Table 3. Again, these results suggest RS method yields more proved robust controllers.

EXAMPLE 3. *The following example is taken from [6]. The system has 3 continuous variables with 4 different modes as follows:*

Table 3: Results for Ex. 2 using SVR method. Itr **: # iterations, Time : total computation time (seconds).**

ϵ	r_D	Itr	Time	Status
0.1	0.01	3	30.4	✓
0.2	0.03	15	87.6	✓
0.2	0.04	15	87.8	✖
0.3	0.0	35	290.9	✖

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = A_q \begin{bmatrix} x \\ y \\ z \end{bmatrix} + B_q + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix},$$

$$A_{q_1} = \begin{bmatrix} 4.15 & -1.06 & -6.7 \\ 5.74 & 4.78 & -4.68 \\ 26.38 & -6.38 & -8.29 \end{bmatrix}, \quad B_{q_1} = \begin{bmatrix} 1 \\ -4 \\ 1 \end{bmatrix},$$

$$A_{q_2} = \begin{bmatrix} -3.2 & -7.6 & -2 \\ 0.9 & 1.2 & -1 \\ 1 & 6 & 5 \end{bmatrix}, \quad B_{q_2} = \begin{bmatrix} 4 \\ -2 \\ -1 \end{bmatrix},$$

$$A_{q_3} = \begin{bmatrix} 5.75 & -16.48 & -2.41 \\ 9.51 & -9.49 & 19.55 \\ 16.19 & 4.64 & 14.05 \end{bmatrix}, \quad B_{q_3} = \begin{bmatrix} -2 \\ 1 \\ -1 \end{bmatrix},$$

$$A_{q_4} = \begin{bmatrix} -12.38 & 18.42 & 0.54 \\ -11.9 & 3.24 & -16.32 \\ -26.5 & -8.64 & -16.6 \end{bmatrix}, \quad B_{q_4} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}.$$

The problem is instantiated for the following parameters; $r_G = 0.1$, $r_I = 0.5$, $r_S = 1.0$. *Again, the goal is to find the most robust controller. The RS method can find a RCLF with disturbance* $r_D = 0.1$ *when* $\epsilon = 0.01$ *is used. The SVR method failed to synthesize a controller for* $\epsilon = 0.3$ *and using* $\epsilon = 0.2$*, it failed to verify the controller for* $r_D = 0.009$*. The controller could guarantee robustness for* $r_D = 0.008$.

EXAMPLE 4. *This benchmark, which includes* 5 *problem instances, is adopted from [19, 22]. The goal is to keep different rooms of an apartment warm, using few number of active heaters. The reader can refer to the mentioned articles for details description of these systems. While these examples do not have disturbances, we incorporate disturbances of the form* $\dot{\mathbf{x}} = f_q(\mathbf{x}) + \mathbf{d}$*, where* f_q *are the original vector fields described in these references. We use these problem instances to demonstrate the scalability. The results for both methods are shown in Table 4. These results demonstrate our method is scalable to larger problems while dealing with robustness. Notice that both methods fail for the last problem instance as the verification of such big problem even using LMI relaxation is expensive.*

5. CONTROLLER SYNTHESIS

Thus far, we have discussed the RCLF certificates and their synthesis using the CEGIS procedure. We present the control implementation in this section. Given a RCLF, the controller is designed according to the feedback law from Eq. (6). Also, according to Lemma 2, we define a minimum dwell time δ between two mode changes triggered by the feedback law above. We consider a periodic (time-triggered) scheme in this section and analyze conditions on the period τ of the feedback law.

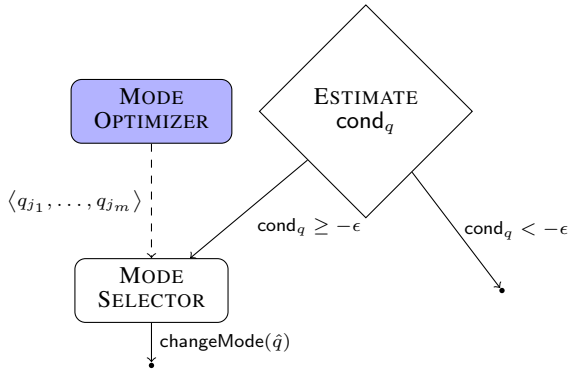


Figure 4: Flow diagram for the controller design showing the various blocks that execute.

Figure 4 demonstrates the overall flow for the periodic task of calculating the feedback from Eq. (6). The first step is the ESTIMATION of $\mathsf{cond}_q(\mathbf{x})$ on the measured state $\mathbf{x}$. If $\mathsf{cond}_q(\mathbf{x}) < -\epsilon$, the controller simply waits one period without changing the control mode. Otherwise, a MODE SELECTOR component is invoked that analyzes a sequence of possible modes to determine a new mode $\hat{q}$. The command to change to this mode is then issued. An optional MODE OPTIMIZER can compute a prioritized sequence of modes as a function of the current state and mode of the plant.

The scheme involves a series of offline computations to determine the key parameters of the controller. Once these computations are performed, the controller performs a series of online computations for each cycle. We start with the online computations needed and discuss the constraints on the periodicity of the feedback law computation.

Rapid Estimation of cond_q**:** Note that $\mathsf{cond}_q(\mathbf{x})$ is defined in Eq. (5) and recalled below:

$$\mathsf{cond}_q(\mathbf{x}) = \max_{\mathbf{d} \in D} \underbrace{(\nabla V).f_q^D(\mathbf{x}, \mathbf{d})}_{g_q(\mathbf{x}, \mathbf{d})}.$$

Since V is computed offline, we may also compute the expression $\nabla V(\mathbf{x})$, and $f_q^D(\mathbf{x}, \mathbf{d})$ for each mode $q \in Q$. The measured value of $\mathbf{x}$ is used and we are required to now compute $\max_{\mathbf{d} \in D} g_q(\mathbf{x}, \mathbf{d})$. This is an instance of a polynomial minimization problem and is very hard to solve precisely in real time. One solution is to compute the feedback function offline [9, 5]. We provide another solution: rather than precisely compute cond_q, we will estimate an over approximation. As an offline step, we wish to choose a nominal disturbance value $\mathbf{d}^*$ and approximate $(\nabla V) \cdot f_q^D(\mathbf{x}, \mathbf{d}) \simeq (\nabla V) \cdot f_q^D(\mathbf{x}, \mathbf{d}^*) + \hat{\mathbf{d}}$, where $\hat{\mathbf{d}} \in \hat{D}$ measures the maximum approximation error possible. Doing so abstracts the plant model and it is possible to run the CEGIS procedure and synthesize an RCLF for this simpler abstract model.

Formally, the error $\hat{D}$ ensures that

$$\begin{aligned} &(\forall\, \mathbf{x} \in S \setminus G)\ (\forall\, \mathbf{d} \in D)\ (\exists \hat{\mathbf{d}} \in \hat{D}) \\ &\nabla V.f_q^D(\mathbf{x}, \mathbf{d}) = (\nabla V(\mathbf{x})).(f_q^D(\mathbf{x}, \mathbf{d}^*) + \hat{\mathbf{d}})\,. \end{aligned} \tag{14}$$

This can be computed/checked offline for a given V and $\mathbf{d}^*$.

Now we redefine $\widehat{\mathsf{cond}}_q(\mathbf{x})$ as

$$\widehat{\mathsf{cond}}_q(\mathbf{x}) : (\nabla V).f_q^D(\mathbf{x}, \mathbf{d}^*) + \max_{\hat{\mathbf{d}} \in \hat{D}} (\nabla V).\hat{\mathbf{d}}\,.$$

The maximization simply involves that of a linear combination of

Table 4: Results for Ex. 4. n **: # state variables,** m**: # modes, Time : total time (seconds), OM: Out of Memory**

Problem		RS					SVR				
n	m	r_D	ϵ	Itr	Time	Status	r_D	ϵ	Itr	Time	Status
3	4	0.04	0.0001	1	25.1	✓	0.005	0.02	4	65.6	✓
							0.006	0.02	4	54.8	✖
							0.0	0.03	4	49.0	✖
4	5	0.02	0.0001	1	155.6	✓	0.001	0.01	4	237.0	✓
							0.002	0.01	4	159.5	✖
							0.0	0.02	6	71.9	✖
5	6	0.001	0.0001	1	1500.3	✓	0.0002	0.002	4	2243.5	✓
							0.0003	0.002	4	872.5	✖
							0.0	0.003	3	89.7	✖
6	4	0.01	0.0001	1	9224.3	✓	0.001	0.01	4	11559.4	✓
							0.002	0.01	4	4237.4	✖
							0.0	0.02	4	333.5	✖
9	4	0.01	0.0001	-	-	OM	0.001	0.01	-	-	OM

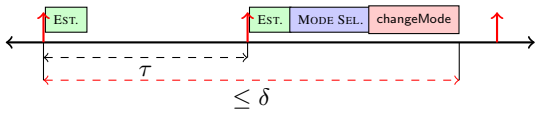
$\hat{\mathbf{d}}$ over a box $\hat{D}$. This is computed online by either selecting the lower bound or upper bound according to the sign of $(\nabla V(\mathbf{x}))_i$.

LEMMA 3. *For each state* $\mathbf{x} \in S$, $\widehat{\mathsf{cond}}_q(\mathbf{x}) \geq \mathsf{cond}_q(\mathbf{x})$.

The approximation error can be reduced arbitrarily (if needed) by subdividing S into multiple regions and choosing a different nominal point $\mathbf{d}^*$ for each region.

Mode Selection: Mode selection considers each possible mode $q_i \in Q$ in turn, optionally according to a prioritized list selected by the optimizer. For each mode it calculates $\widehat{\mathsf{cond}}_{q_i}(\mathbf{x})$ and selects the first mode for which $\widehat{\mathsf{cond}}_{q_i}(\mathbf{x}) < -\epsilon$.

Task Schedule: We now analyze the task schedule. Our analysis assumes that the worst case execution times for various components are known. In particular, let w_e and w_{ms} be the worst case times for the ESTIMATION and MODE SELECTION, respectively. We will assume for simplicity that (a) the states $\mathbf{x}$ are measured/estimated in parallel to provide a new update each time the feedback task is invoked, and (b) the command for changing mode takes effect within at most w_c time after the command is issued.



In the worst case, we require that the time period τ be large enough so that the ESTIMATION, MODE SELECTION and changeMode all run in a single period. I.e, $\tau > w_e + w_{ms} + w_c$. Also, we require that if the event $\mathsf{cond}_q(\mathbf{x}) \geq -\epsilon$ happens just after a feedback computation has commenced, then in the worst case, the time taken to notice the change and react to it be shorter than the min dwell time δ. I.e, $\tau + w_e + w_c + w_{ms} \leq \delta$. Combining, we obtain the constraints

$$w_e + w_{ms} + w_c < \tau \leq \delta - w_e - w_c - w_{ms}\,. \qquad (15)$$

Note that under situations where the controller does not switch, there is idle time. This time could presumably be used to run optional computations such as that of the optimizer, which selects an appropriate sequence of prioritized modes.

Finally, our design requires the computation of feedback law periodically at each time τ. It is, in fact, possible to defer the computation of the feedback law based on the current state $\mathbf{x}$. From the proof of Lemma 1, we derive a bound

$$|\mathsf{cond}_q(\mathbf{x}(T+t)) - \mathsf{cond}_q(\mathbf{x}(T))| \leq \Lambda_q t\,,$$

wherein Λ_q is computed offline as a function of the safe set S and the RCLF V. Suppose, we have an estimate for $\mathsf{cond}_q(\mathbf{x}(T))$ at time T and it is less than $-\epsilon$, we can in fact conservatively estimate a future time $T+t$ at which $\mathsf{cond}_q(\mathbf{x}(T+t)) \geq -\epsilon_s$ as $t = \frac{\epsilon_s + \widehat{\mathsf{cond}}_q(\mathbf{x}(T))}{\Lambda_q}$. In practice, it may be the case that $t \gg \tau$, which allows us to avoid unnecessary recomputation of the feedback law.

Offline Computations: We now summarize the offline calculations that will be needed to design a controller. First and foremost, a template for V is chosen. Then the modules in Figure 4 are synthesized and a WCET estimation is used to predict their WCETs statically, yielding w_{ms}, w_e and w_c. This allows us to design the period τ. Only then, the control disturbance is calculated to model delays caused by WCETs. Then, we require an estimation of $\hat{D}$ for a disturbance estimate around a nominal point, that allows us to write $\nabla V \cdot f_q^D(\mathbf{x}, \mathbf{d})$ for each mode as $\nabla V \cdot f_q^D(\mathbf{x}, \mathbf{d}^*) + \hat{\mathbf{d}}$. This model is input to a RCLF synthesis tool to generate coefficients of V. After acquiring V, offline computations are needed to compute the minimum dwell time δ and check Equations (14) and (15).

5.1 Two Case Studies

Inverted Pendulum For the first case study, a classical inverted pendulum example is considered. The system of interest has two state variables θ and ω with the following dynamics

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \omega \\ 4.9\sin(\theta) - 4\omega + 2\cos(\theta)u \end{bmatrix},$$

where u is the control input belong to set $U : [-30, 30]$. The region of interest is $S : \{[\theta\ \omega]^T | \theta^2 \leq 1.5^2, \omega^2 \leq 4^2\}$. The goal is to reach region $G : \{[\theta, \omega]^T | \theta^2 + \omega^2 \leq 0.2^2\}$, starting from region $I : \{[\theta, \omega]^T | \theta^2 + \omega^2 \leq 0.5^2\}$. Trigonometric functions are approximated by degree 4 polynomials. and the plant disturbance $\mathbf{d} \in D^P : [-0.02, 0.02]^2$. The modified dynamics are given by

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \omega \\ \begin{pmatrix} 4.9(\theta - 0.1667\theta^3) - 4\omega \\ +2(1 - \omega^2 + 0.0417\omega^4)u \end{pmatrix} \end{bmatrix} + \begin{bmatrix} d_x^P \\ d_y^P \end{bmatrix}.$$

The set U is discretized to the set $\mathfrak{U} : \{-30, 30\}$ to yield a switch system. Notice that the control implementation can always choose a larger set of modes. For example, control can allow all u s.t.

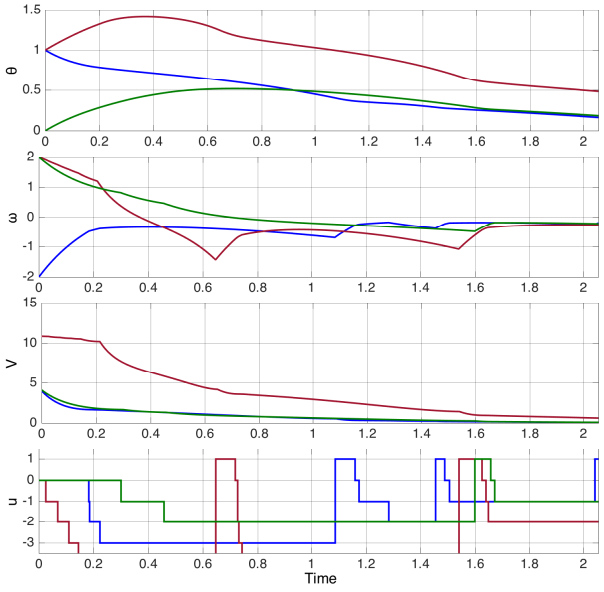


Figure 5: Simulation of the Closed Loop System for Inverted Pendulum. States, Lyapunov function and control input are shown for each trace with different colors.

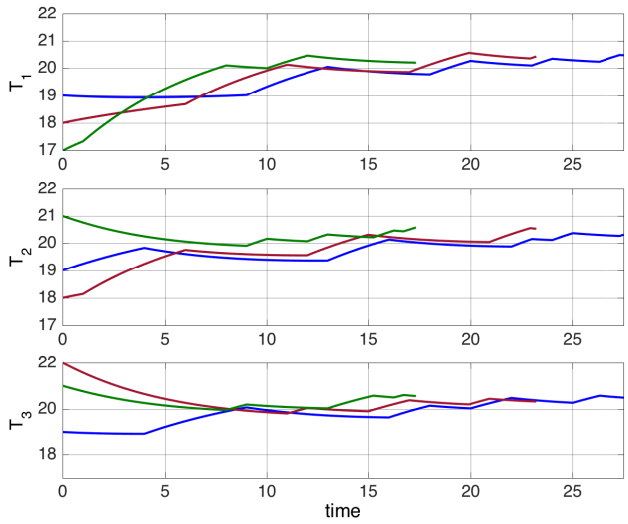


Figure 6: Simulation of the Closed Loop System for Heater Example. Temperatures are shown for each trace with different colors.

$|u| \in \{0, 0.5, 1, 2, 3, 6, 30\}$. Also, the measurement error is set to 0.0001. Let the worst case execution times are $w_{ms} = w_e = 0.5\mu s$ and w_c is ignored $w_c = 0$. This yields $\tau = 1\mu s$. Then, all the uncertainties in controller are modeled by $e = \mathbf{x} + \mathbf{d}^C$ with the control disturbance space $D^C : [-2 \times 10^{-4}, 2 \times 10^{-4}]^2$ (measurement error and time delay as result of computation time). Next, set $\hat{D}$ is estimated as $\hat{D} : [-0.1, 0.1]^2$. We use $\epsilon = 0.02$ and we find a RCLF (V) using CEGIS procedure (within 5 iterations):

$$V(\theta, \omega) = 3.3163338\,\theta^2 + 1.7209759\,\theta\omega + 1.0273113\,\omega^2\,.$$

Calculated δ is $2\mu s$ using ($\epsilon_s = 0.002$). We also have Equations (14) and (15) checked. We implemented the plant and controller in Matlab(tm) using the Simulink(tm) design environment. Some traces of the system are shown in Fig. 5 for three different initial states (using three different color) with some uniformly random control disturbances. Fig. 5 shows values of states, RCLF V and control input through the time. Each simulation is stopped when G is reached. As shown in the figure, the minimum switch time in the simulation is 0.005 which is far bigger than the calculated δ.

Room Heating The second case study is the first problem instance from Example 4. The goal is to control the temperature of three rooms (T_1, T_2 and T_3) and keep them around 21. There are four different modes. Either the heater is off or the heater is on in at most one of the rooms. The dynamics are described below

$$\begin{bmatrix} \dot{T_1} \\ \dot{T_2} \\ \dot{T_3} \end{bmatrix} = \begin{bmatrix} -0.105T_1 + 0.05T_2 + 0.05T_3 + 0.05 \\ 0.05T_1 - 0.105T_2 + 0.05T_3 + 0.05 \\ 0.05T_1 + 0.05T_2 - 0.105T_3 + 0.05 \end{bmatrix} + B_q,$$

$$B_{q_0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; B_{q_1} = \begin{bmatrix} 0.5 - 0.01T_1 \\ 0 \\ 0 \end{bmatrix}$$

$$B_{q_2} = \begin{bmatrix} 0 \\ 0.5 - 0.01T_2 \\ 0 \end{bmatrix}, \; B_{q_3} = \begin{bmatrix} 0 \\ 0 \\ 0.5 - 0.01T_3 \end{bmatrix}.$$

In this example, the worst case execution times are $w_{wd} = w_e = 0.004$ and $w_c = 0.005$. We assume e has form $\mathbf{x} + \mathbf{d}^C$ and control disturbance is in $[-0.01, 0.01]^3$ models both measurement error and delay cause by execution time. Also, we assume the plant has disturbance $[-0.01, 0.01]^3$, meaning that temperature of each room can change by disturbance, at rate $1^\circ C$ per 100 seconds. To use the estimated disturbance model, we choose $\hat{D} : [-0.04, 0.04]^3$. The RCLF (which is calculated in Example 4 by $\epsilon = 0.001$) is used and δ is set to be 0.02. Next, it is confirmed that $\hat{D}$ and δ are big enough by checking Equations (14) and (15).

The optimizer uses a MPC paradigm for prioritizing modes for selection according to their costs. The cost function considers (a) switch cost, (b) operation cost and (c) terminal cost. Switch cost is 1 if a switch is needed and operation cost for mode q is 1 if a heater is on for mode q. Also, terminal cost for mode q and optimized time T_q^* is $\frac{V(\mathbf{x}_q(T_q^*))}{T_q^*}$, where $\mathbf{x}_q(.)$ shows trace of $\mathbf{x}$ under dynamics of mode q.

The plant and controller are implemented in Simulink(tm) design environment. Some traces for different initial states are shown in Fig. 6. T_i is temperature for room i and simulations are stopped, once set G is reached.

6. CONCLUSIONS

In this paper, robust controller synthesis using RCLFs is considered. RCLFs guarantee that there is a switching strategy even in presence of disturbances. We provided a CEGIS framework to generate RCLFs automatically. We demonstrated that using RCLF to synthesis controller gives provably more robust controllers compare to cases when only some parameters are tuned for increasing robustness. Next, we showed that under certain disturbance models, the controller can implement the switching strategy efficiently.

For future work, we wish to investigate problem of synthesizing output feedback controllers and focus on a larger class of temporal properties.

7. ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation (NSF) under CNS-0953941 and CCF-1527075. All opinions expressed are those of the authors and not necessarily of the NSF.

8. REFERENCES

[1] R. Alur, R. Bodik, G. Juniwal, M. M. Martin, M. Raghothaman, S. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, A. Udupa, et al. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 1–8. IEEE, 2013.

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

[3] E. Aydin Gol, M. Lazar, and C. Belta. Language-guided controller synthesis for discrete-time linear systems. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 95–104. ACM, 2012.

[4] S. Battilotti. Universal controllers for robust control problems. *Mathematics of Control, Signals and Systems*, 10(2):188–202, 1997.

[5] S. Battilotti. Robust stabilization of nonlinear systems with pointwise norm-bounded uncertainties: A control lyapunov function approach. *IEEE transactions on automatic control*, 44(1):3–17, 1999.

[6] P. Bolzern and W. Spinelli. Quadratic stabilization of a switched affine system about a nonequilibrium point. In *American Control Conference, 2004. Proceedings of the 2004*, volume 5, pages 3890–3895. IEEE, 2004.

[7] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[8] R. Dimitrova and R. Majumdar. Deductive control synthesis for alternating-time logics. In *Embedded Software (EMSOFT), 2014 International Conference on*, pages 1–10. IEEE, 2014.

[9] R. A. Freeman and P. Kokotovic. Inverse optimality in robust stabilization. *SIAM journal on control and optimization*, 34(4):1365–1391, 1996.

[10] G. Frehse, S. K. Jha, and B. H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *Hybrid Systems: Computation and Control*, pages 187–200. Springer, 2008.

[11] S. Gao, S. Kong, and E. M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *Intl. Conference on Automated Deduction (CADE)*, pages 208–214, 2013.

[12] L. E. Ghaoui and V. Balakrishnan. Synthesis of fixed-structure controllers via numerical optimization. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 3, pages 2678–2683. IEEE, 1994.

[13] D. Henrion, J.-B. Lasserre, and J. Löfberg. Gloptipoly 3: moments, optimization and semidefinite programming. *Optimization Methods & Software*, 24(4-5):761–779, 2009.

[14] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri. Controller synthesis with inductive proofs for piecewise linear systems: an smt-based algorithm. *arXiv preprint arXiv:1509.04623*, 2015.

[15] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Aréchiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142. ACM, 2014.

[16] J. Liu, N. Ozay, U. Topcu, and R. M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 58(7):1771–1785, 2013.

[17] M. Mazo Jr, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *Computer Aided Verification*, pages 566–569. Springer, 2010.

[18] A. Mosek. The mosek optimization software. *Online at http://www.mosek.com*, 54, 2010.

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

[20] P. Nilsson and N. Ozay. Incremental synthesis of switching protocols via abstraction refinement. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014.

[21] N. Ozay, J. Liu, P. Prabhakar, and R. M. Murray. Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems. In *American Control Conference (ACC), 2013*, pages 6237–6244. IEEE, 2013.

[22] H. Ravanbakhsh and S. Sankaranarayanan. Counter-example guided synthesis of control lyapunov functions for switched systems. In *Decision and Control (CDC), 2015 IEEE 54rd Annual Conference on*. IEEE, 2015.

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

[24] M. A. Rotea and P. P. Khargonekar. Stabilization of uncertain systems with norm bounded uncertainty-a control lyapunov function approach. *SIAM Journal on Control and Optimization*, 27(6):1462–1476, 1989.

[25] A. Solar-Lezama. *Program synthesis by sketching*. ProQuest, 2008. PhD thesis (University of California, Berkeley).

[26] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat. Combinatorial sketching for finite programs. In *ACM Sigplan Notices*, volume 41, pages 404–415. ACM, 2006.

[27] E. D. Sontag. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13(2):117–123, 1989.

[28] A. Taly and A. Tiwari. Switching logic synthesis for reachability. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 19–28. ACM, 2010.

[29] W. Tan and A. Packard. Searching for control lyapunov functions using sums of squares programming. *sibi*, 1:1, 2004.

[30] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 313–314. ACM, 2011.

[31] B. Yordanov and C. Belta. Parameter synthesis for piecewise affine systems from temporal logic specifications. In *Hybrid Systems: Computation and Control*, pages 542–555. Springer, 2008.