ON SIMULATION BASED VERIFICATION OF NONLINEAR NONDETERMINISTIC HYBRID SYSTEMS

BY

ZHENQI HUANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Adviser:

Assistant Professor Sayan Mitra

ABSTRACT

Automatic safety verification of hybrid systems typically involves computing precise reach sets of such systems. This computation limits scalability of verification as for many model classes it scales exponentially with the number of continuous variables. First we propose a simulation-based algorithm for computing the reach set of a class of deterministic hybrid system. The algorithm first constructs a cover of the initial set of the hybrid system. Then the reach set of executions from the same cover are overapproximated by simulation traces and tubes around them. Experiments are performed on several benchmark problems including navigation benchmarks, room heating benchmarks, non-linear satellite systems and engine hybrid control systems. The results suggest the algorithm may scale to larger systems. Finally, we present a reachability algorithm that computes precise reach set of dynamical systems $\mathcal{A}$ with non-linear differential inclusions. The algorithm constructs a sequence of shrink concretizations of $\mathcal{A}$. Then the reach sets of the concretizations are used to construct an overapproximation of the reach set of $\mathcal{A}$. Soundness and Completeness of both algorithms presented are formally proved.

To my parents for their love and support and
To my late grandfathers for their concern

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Sayan Mitra, who makes this thesis possible. Not only because of his enthusiasm in research and patience in teach but also his nice character as a mentor and a friend. I also want to thank my colleagues - Taylor Johnson, Sridhar Duggirala, Stanly Bak, Adel Ahmadyan, Jeremy Green, Fardin Abdi and others. We are good friends who are able to exchange idea on research and outside research. This research was partially funded by grants from John Deere and National Science Foundation Grant CNS 10-16791.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Computing systems that control physical processes have become commonplace in safety-critical applications such as vehicle control, smart grid, avionics, air traffic control and industrial automation. Design and operation of these *hybrid systems* should provide a high level of assurance because failures and misbehaviors can lead to significant financial, environmental, and societal losses. A recent example was the recall of Volkswagen because of problems associated with the automated DSG gear boxes in 2012, including excessive shift shock, unable to shift gear and abnormal increase in engine rpm. The recall affects approximately 500,000 vehicles in China and a upgrade of software is issued to resolve the problem [1]. Another example of failure of hybrid system is the north America blackout in 2003, which dues to race condition of the monitoring software [2]. The black out is the second most widespread blackout in the history, which affected an estimated 10 million people in Ontario and 45 million people in 8 U.S. states.

Although high assurance is a desirable goal, as the systems become more sophisticated, it is becoming more and more challenging to attain using standard design methodologies. The standard technique for detecting design flaws is based on testing and simulations. Since the correctness of these systems crucially depends on the complex interactions between the computing and physical elements, it is difficult and in some cases it is impossible to cover the set of all possible behaviors with a finite number of simulations or tests. Furthermore, the simulation tools often lack precise semantics. This makes it impossible to arrive at rigorous assurance guarantees from the simulation runs.

Formal verification provides an alternative to these approaches. This approach involves first building a mathematical model for the system (with precise semantics), and then using deductive or algorithmic techniques to establish properties about *all* possible behaviors of the model in one go. The most common algorithmic approach involves computing the set of reachable states of the model. This enables one to check if any of those reachable states are unsafe. Algorithmic verification has seen success in establishing correctness of many hardware and software systems, and based on recent developments, they present an attractive approach for verification of embedded control systems (see [3] for a review).

A middle road between these two approaches has recently gained traction in the hardware and software communities [4, 5]. In this approach, simulation traces are used for providing system-level formal reliability guarantees by appropriately generalizing individual traces to a set of executions and then verifying the property for this set. Developing a "verification from execution" approach for embedded and cyber-physical systems presents several challenges. First and foremost, these systems evolve both discretely and continuously over time but in a simulation trace the states can be observed and recorded only at discrete points in time. The gaps in the trace have to be filled somehow for reasoning about the actual execution. Secondly, the recorded states in an execution may be different from actual states that are visited in an execution because of quantizations and numerical integration. In this thesis, we propose solutions to these problems, which compute tubes around simulation traces guarantee to contain all execution of the system. In the computation, formal models of the systems are analyzed to extract a metric of distance between executions start from initial states close to each other.

1.1 Overview of Contributions

Consider a deterministic hybrid automaton $\mathcal{A}$ (see [6, 7]) with set of initial states Θ. A trace β from $\mathbf{x}_0 \in \Theta$ is a sequence of, possibly inaccurate, samples of an execution of $\mathcal{A}$. In this thesis, we show the following: given a deterministic hybrid system $\mathcal{A}$ with initial set Θ, there exists some finite cover of Θ, which we call (δ, γ, k)-*representative cover*, such that the radius of each cover is bounded by γ and the executions start in the same cover visit the same sequence of locations evaluated in the first k δ-samples. We assume that such (δ, γ, k)-representative covers can always be constructed by some oracle. Then, given a simulation trace β from a state in a cover and a hybrid model $\mathcal{A}$ of the system, we can compute an overapproximation of the time-bounded reachable states of any execution of $\mathcal{A}$ that may have produced the trace β from initial states in the same cover as β. This enables us to verify bounded time safety property. The proposed algorithm combines the dynamic information from β and the static information from the model $\mathcal{A}$.

We have implemented a MATLAB tool called *Hybrid Trace Verifier (HTV)* that implments this methodology and uses traces generated from Mathwork's Simulink/Stateflow (SLSF) [8]—a popular modeling environment for embedded systems. For static analysis HTV translates the SLSF models to hybrid automata [9, 7] using HyLink [10].

Building-up on the algorithm for deterministic HA, we develop algorithms that can verify models $\mathcal{A}$ specified by differential inclusions. We introduce *shrink concretizations* of $\mathcal{A}$, which are obtained in shrink $\mathcal{A}$'s initial set as well as the differential inclusion specifies $\mathcal{A}$. With this notion, systems $\mathcal{A}$ can be concretized to a system with differential equations $\mathcal{B}$, whose reach set is in general easier to compute compared to $\mathcal{A}$'s. Our analysis establishes that the reach set of $\mathcal{A}$ can be upper bounded by bloating $\mathsf{Reach}_{\mathcal{B}}$ (the reach set of $\mathcal{B}$) by a factor of M_1 and at the same time be lower bounded by bloating $\mathsf{Reach}_{\mathcal{B}}$ by a factor of M_2. With such relation, we can manipulate the reach set of $\mathcal{B}$, which is in general easier to compute, to derive tight overapproximations of the reach set of $\mathcal{A}$. We formally prove the soundness and completeness of our algorithms.

1.2 Related Work

There is a large and growing body of work on automatic safety verification of cyber-physical systems modeled as different types of hybrid systems. Initially several classes of hybrid systems were identified for which the exact reachability problem is decidable. These classes include timed automata [11] and rectangular initialized hybrid automata [12, 13]. However only a restricted class of piratical systems can be modeled by these classes of hybrid automata, subsequently , practical algorithms and data structures for computing the reach sets of more general hybrid systems are studied. Several useful types of data-structures have been proposed including polyhedra [14], zonotopes [15], ellipsoids [16] and semi-algebraic sets [17]. Approximate reachability algorithms based on these data-structures are embodied in tools such as HyTech [18], Checkmate [19], PHAVer [20], and more recently SpaceEx [21].

Several recent papers address the problem of formally verifying hybrid systems from simulation traces [22, 23]. The approach presented in [23] generates a symbolic trace from a given simulation trace by carefully instrumenting SLSF models. The symbolic traces are then used to compute a set of initial states that from which all traces visit the same sequence of locations. Such instrumentation can be used to generate simulation traces for the method proposed in this thesis. The approach presented in [24] and [22] searches for counterexamples to Metric Temporal Logic (MTL) properties for (possibly non-linear) hybrid systems through minimization of a robustness metric. The global minimization is carried out using Monte-Carlo techniques. Guaranteed ODE solvers have been developed to compute error bounds on the solutions of differential equations. For example, the C++ library in [25] solves an initial value problem and computes a discrete sequence of error terms which upper bound the distances to the true solution at each sampled point.

Several existing algorithms focus on hybrid system specified with rectangular dynamics [26, 27] or differential equations [18, 20, 28]. More recently, reachability algorithms for hybrid system specified with linear differential inclusions are introduced [16, 29, 30]. The algorithms presented in [29] employs a data structure where the reach set of hybrid system are overapproximated by unions and intersections of ellipsoids. Then the reach set is computed by propagating and overapproximating the ellipsoids with linear differential inclusion in a discrete time fashion. The algorithm in [30] analyzes the vector fields on the state space generated by linear differential inclusions. Invariants of the reach set can be generated by searching the state space.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 presents necessary background information and the hybrid automaton modeling framework that is used throughout the thesis. Chapter 3 develops the reachability algorithms that compute an overapproximation of the reach set of a deterministic non-linear hybrid automaton $\mathcal{A}$. We also show that the overapproximation our algorithm computes can be made arbitrarily tight. Then, in Chapter 4 we introduce the implementation of the tool **HTV** and present experiment results on hybrid system benchmarks upto 10 continuous variables, including room heating systems, navigation systems, satellites systems and engine control systems. Finally, in Chapter 5 we present an algorithm computes the overapproximation of the reach set of a dynamical system $\mathcal{A}$ upto arbitrary accuracy, where the trajectories of $\mathcal{A}$ are specified by a set of differential inclusions.

CHAPTER 2

PRELIMINARIES

2.1 Functions, Sets, and Set-valued Functions

We will introduce the hybrid I/O automaton modeling framework, but first we begin with some notation for functions and sets. Without specification, a function is single-valued. For any function $f : A \mapsto B$ and a subset set of its domain $S \subseteq A$, we write the *restriction* of f to S as $f \lceil S : S \mapsto B$ such that $f \lceil S(x) = f(x)$ for each $x \in S$. For a function f whose range is a set of functions, we write $f \downarrow S$ such that for each $x \in A$, $f \downarrow S(x) = f(x) \lceil S$.

A function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is *smooth* if all its higher derivatives exist. It is said to be *Lipschitz* if there exists $L > 0$ such that $|f(x) - f(y)| \leq L|x - y|$ for all $x, y \in \mathbb{R}^n$.

For a vector $x \in \mathbb{R}^n$, $|x|_p$ denotes the ℓ^p norm. For a matrix $A \in \mathbb{R}^{mn}$, $|A|_p$ denotes the induced p norm of A, that is $|A|_p \triangleq \sup\limits_{|x|_p=1} |Ax|_p$. Without a subscript, $|x|$ and A denote the ℓ^∞ norm and induced infinity norm. $B_\delta(x) \subseteq \mathbb{R}^n$ denotes a *closed* ball with radius $\delta > 0$ around the point $x \in \mathbb{R}^n$ measured in infinity norm.

For a set $S \subseteq \mathbb{R}^n$, we denote S^o, S^c and ∂S as the interior, closure and boundary of S respectively. Formally, $S^o = \{x \in \mathbb{R}^n : \exists \delta > 0, B_\delta(x) \subseteq S\}$, $S^c = \{x \in \mathbb{R}^n : \forall \delta > 0, B_\delta(x) \cap S \neq \emptyset\}$, and $\partial S = \{x \in S^c : x \notin S^o\}$.

For two set $X, Y \subseteq \mathbb{R}^n$, $d(X, Y)$ denotes the Hausdorff distance, that is, $d(X, Y) \triangleq \max\{\sup\limits_{x \in X} \inf\limits_{y \in Y} |x - y|, \sup\limits_{y \in Y} \inf\limits_{x \in X} |x - y|\}$. The diameter of a set $S \subseteq \mathbb{R}^n$ is the supremum distance between its any two elements, that is $D(S) \triangleq \sup_{x,y \in S} |x - y|$. The compact set $S \subseteq \mathbb{R}^n$ expanded by $\gamma \in \mathbb{R}_{\geq 0}$ is defined as $\mathsf{Expand}(S, \gamma) \triangleq \bigcup\limits_{x \in S} B_\gamma(x)$. We define $\mathsf{Shrink}(S, \gamma) \triangleq \{x | x \in S \wedge (\min\limits_{y \in \partial S} |y - x|) \geq \gamma\}$ as the set S shrunk by γ. For a point $x \in \mathbb{R}^n$ and some subset $A \subseteq \mathbb{R}^n$, we define $\mathsf{Proj}_A(x) \triangleq \operatorname*{argmin}\limits_{y \in A} |x - y|$ as the Euclidean projection of x onto A. That is, the set of elements in set A that are closest to the point x. For a set A, a collection of points $\{x_i\}_{i=1}^N$ is a *δ-cover* of A if $A \subseteq \cup_{i=1}^N B_\delta(x_i)$. That is, the union of the delta balls around x_i covers the set A.

A *set-valued function*, also known as multifunction or correspondence, is a relation, where each input is associated with one or more than one outputs. For example, $F : \mathbb{R}^n \mapsto 2^{\mathbb{R}^m}$ maps each point in $\mathbb{R}^n$ to a subset of the set $\mathbb{R}^m$.

A set-valued function $F : \mathbb{R}^n \mapsto 2^{\mathbb{R}^m}$ is said to be *upper hemicontinuous* at $x \in \mathbb{R}^n$ if for every open set $O \supseteq F(x)$, there exists a $\delta > 0$ such that for all $y \in B_\delta^o(x)$, $F(y) \subseteq O$. F is said to be *lower hemicontinuous* at $x \in \mathbb{R}^n$ if for every open set O such that $O \cap F(x) \neq \emptyset$, there exists a $\delta > 0$ such that for all $y \in B_\delta^o(x)$, $O \cap F(x) \neq \emptyset$. F is said to be *continuous* if it is both upper hemicontinuous and lower hemicontinuous. A set-valued function F is said to be Lipschitz with a Lipschitz constant $L > 0$ if for any $x, y \in \mathbb{R}^n$, $d(F(x), F(y)) \leq L|x - y|$.

A single-valued function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is said to be a *selection* of set-valued function F if for any $x \in \mathbb{R}^n$, $f(x) \in F(x)$. If F is lower hemicontinuous and its value is always closed and convex, then there exists a continuous selection f of F ([31]). Moreover, If F is Lipschitz with constant L, its selection f is also Lipschitz with constant L.

2.2 Variables and Trajectories

A *variable* is a name used to identify state components of automata and communication channels between automata. Each variable v is associated with a type, *type*(v), which is the set of values v can take. A *valuation* for a set of variables V, maps each $v \in V$ with a value in *type*(v). For a set of variables V, *val*(V) denotes the set of all possible valuations of V. Valuations are denoted by $\mathbf{v}, \mathbf{x}, \mathbf{x}'$, etc. For a valuation $\mathbf{v}$ of V the value of a variable $v \in V$, is denoted by $\mathbf{v}.v \triangleq \mathbf{v} \lceil \{v\}$.

A *trajectory* ξ for V is a function $\xi : [0, t] \to Q$, where $t \in \mathbb{R}_{\geq 0}$. That is, a trajectory maps a time value in $[0, t]$ to a valuations of V. We define $\xi.\mathsf{fstate} \triangleq \xi(0)$, $\xi.\mathsf{lstate} \triangleq \xi(t)$ and $\xi.\mathsf{dur} \triangleq t$. A variable is *continuous* if all its trajectories are piecewise-continuous. A *discrete* variable is a special type of continuous variable whose trajectories are piece-wise constant. Given a trajectory ξ of V, the restriction of the trajectory to a variable $v \in V$ is denoted by $\xi \downarrow v$. The concatenation of two trajectories ξ_1, ξ_2 is done by taking the union between the first trajectory and the second where the second trajectory has its domain shifted by the limit time of the first. That is, let $\xi \triangleq \xi_1 \frown \xi_2$, then (i) $\xi(t) = \xi_1(t)$ for $t \in [0, \xi_1.dur)$, and (ii) $\xi(t) = \xi_2(t - \xi_1.dur)$ for $t \in [\xi_1.dur, \xi_1.dur + \xi_2.dur]$.

2.3 Hybrid Automata

The following definition of *hybrid automata* makes a few minor changes to the standard one [6, 7] for a cleaner presentation of the results in this paper.

Definition 1. *A* hybrid automaton *(HA) $\mathcal{A}(\Theta)$ is a tuple $\langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T}\rangle$ where:*

(i) $V = X \cup \{loc\}$ is a finite set of variables, where loc is a discrete variable of finite type $\mathcal{L}$ called the set of locations*, and each $x \in X$ is a continuous real-valued variable. $Q \subseteq val(V)$ is a set of states. $\Theta \subseteq Q$ is a non-empty bounded set of initial states.*

(ii) $Grd : \mathcal{L} \times \mathcal{L} \mapsto 2^{val(X)}$, is a function that maps each pair $i, j \in \mathcal{L}$ to a set in $val(X)$. In addition, we assume that every guard is compact.

(iii) A transition $\mathbf{v} \to \mathbf{v}'$ can occur if $\mathbf{v}.X \in Grd(\mathbf{v}.loc, \mathbf{v}'.loc)$ and $\mathbf{v}.X = \mathbf{v}'.X$, and we write this transition as $\mathbf{v} \to \mathbf{v}'$. That is, the transitions do not change the continuous state.

(iv) $Inv : \mathcal{L} \mapsto 2^{val(X)}$ is a function that maps each location to a set of states, called invariant set.

(v) $\mathcal{T}$ is a set of differentiable trajectories for V which is closed under prefix and suffix. Typically, $\mathcal{T}$ is specified by a set of flow conditions that is a set of differential inclusions.

Our definition of hybrid automaton is similar to the general definition ([7]) with the extra assumption that the continuous variables are not reset during the discrete transitions.

Semantics. In this paper, we discuss hybrid automaton with differential inclusions. For each $l \in \mathcal{L}$, the corresponding F_l is a set-valued function. For each trajectory $\xi \in \mathcal{T}$, (a) the discrete variable loc remains constant along ξ denoted by $\xi.loc \triangleq \xi(0).loc$, (b) for all $t \in [0, \xi.\mathsf{dur})$ the state of ξ lies in invariant: $\xi(t) \in Inv(\xi.loc)$, and (c) for any $t \in [0, \xi.\mathsf{dur})$, the derivative of the continuous variables X along the trajectory satisfy the differential inclusion

$$\frac{d}{dt}\xi(t).X \in F_{\xi.loc}(\xi(t).X). \tag{2.1}$$

In general, starting at a state $\mathbf{v} \in Q$, there may be multiple trajectories with non-zero durations. Specially, if for all $l \in \mathcal{L}$ and all $x \in Inv(l)$, $F_l(x)$ is a singleton, then Equation (2.1) reduces to a differential equation

$$\frac{d}{dt}\xi(t).X = f_{\xi.loc}(\xi(t).X), \tag{2.2}$$

where f_l is the selection of F_l in domain $Inv(l)$. For a HA defined with differential equations, there is a unique trajectory from a state $\mathbf{v} \in Q$.

For an automaton $\mathcal{A}$, we refer to the components of the automaton as $V_{\mathcal{A}}, \mathcal{L}_{\mathcal{A}}, Q_{\mathcal{A}}, \Theta_{\mathcal{A}}$, $Grd_{\mathcal{A}}$, $Inv_{\mathcal{A}}$, $\mathcal{T}_{\mathcal{A}}$ etc.

2.4 Executions, Reach Sets, and Safety

An *execution fragment* α of automaton $\mathcal{A}$ is a sequence of trajectories $\alpha = \xi_0 \frown \xi_1 \frown$, where each $\xi_i \in \mathcal{T}$, and $\xi_{i-1}.\mathsf{lstate} \to \xi_i.\mathsf{fstate}$. The first state of α, $\alpha.\mathsf{fstate}$ is denoted by $\xi_0.\mathsf{fstate}$. If α is a finite sequence ending with a closed trajectory ξ_n, then its last state $\alpha.\mathsf{lstate}$ is defined as $\xi_n.\mathsf{lstate}$ and its duration $\alpha.\mathsf{dur} \triangleq \sum_{i=0}^{n} \tau.\mathsf{ltime}$. Formally, for any time $t \in [0, \alpha.\mathsf{dur}]$, we define the state of $\mathcal{A}$ at time t, denoted by $\alpha(t)$, as $\alpha'.\mathsf{lstate}$ where α' is the longest prefix of α with $\alpha'.\mathsf{dur} \leq t$.

An execution fragment is an *execution* if it starts at an initial state, that is, $\alpha.\mathsf{fstate} \in \Theta$. The set of all executions is denoted by $\mathsf{Execs}_{\mathcal{A}}$. The set of executions and execution fragments up to time T are denoted $\mathsf{Execs}_{\mathcal{A}}^{T}$ and $\mathsf{Frags}_{\mathcal{A}}^{T}$.

A state $\mathbf{v}$ is *reachable* if there exists an execution α with $\alpha.\mathsf{lstate} = \mathbf{v}$. Define $\mathsf{Reach}_{\mathcal{A}}(t_1, t_2) \subseteq Q$ as $\mathbf{v} \in \mathsf{Reach}_{\mathcal{A}}(t_1, t_2)$ iff there exists an execution $\alpha \in \mathsf{Execs}_{\mathcal{A}}$ and a time $t \in [t_1, t_2]$ such that $\alpha(t) = \mathbf{v}$. We write $\mathsf{Reach}_{\mathcal{A}}(t, t)$

simply as $\mathsf{Reach}_{\mathcal{A}}(t)$ and $\mathsf{Reach}_{\mathcal{A}}(0,T)$ as $\mathsf{Reach}_{\mathcal{A}}^{T}$. Given a set of states $\mathcal{U} \subseteq Q$, $\mathcal{A}$ is said to be T-*safe* with respect to $\mathcal{U}$ if $\mathsf{Reach}_{\mathcal{A}}^{T}$ and $\mathcal{U}$ are disjoint. It is said to be *safe* if it is T-safe for all T. Otherwise it is said to be *unsafe*. Thus, the crucial step in verifying safety property of $\mathcal{A}$ is to compute the exact or an accurate approximation of $\mathsf{Reach}_{\mathcal{A}}$.

Several special classes of hybrid automata have been identified for which $\mathsf{Reach}_{\mathcal{A}}$ and $\mathsf{Reach}_{\mathcal{A}}^{T}$ can be computed exactly [11, 27, 32, 33]. For general automata with complex continuous dynamics, exact computation is undecidable and one has to rely on overapproximations for safety verification. Over the past decade, several algorithms and tools have been developed for computing overapproximations of bounded reach sets of different classes of hybrid automata. Examples include tools such as HyTech [18], UPPAAL [34], PHAVer [20], and the more recent SpaceEx [21].

2.5 Simulation Oracles

Besides the reachability tools introduced in Section 2.4, a various of simulation engines have been used to analyze the reliability of hybrid systems. For a hybrid automaton, these engines produce a sampled trace of an individual execution with some guarantees, which can be useful for finding defects. While this method is usually computationally less expensive compared to verification, it suffers from the incompleteness. A middle road between these two approaches has recently gained traction [4, 5]. In this approach, simulation traces are used for providing system-level reliability guarantees by appropriately generalizing individual traces to a set of executions and then verifying the property for this set.

For most of the simulation engines, a simulation trace of a HA, given initial state $\theta \in \Theta$, is computed iteratively as following: starting at a precomputed sample point $\mathbf{v}_{i-1}.loc = l$ i the continuous part of next sample point $\mathbf{v}_i$ is obtained by numerically solving the initial value problem $\dot{x} = f_l(x)$, with initial state $x(0) = \mathbf{v}_{i-1}.X$ for a period of time δ. ii the discrete part of next sample point is determined as: $\mathbf{v}_i.loc = l'$ if $\mathbf{v}_i.X \in Inv(l')$. The iteration continues until the simulation time reaches the bound $T > 0$. Advanced numerical algorithms provides stepwise bound in the computation error. We capture these features of simulation traces in the definition below.

Definition 2. *Fixed a HA $\mathcal{A}(\Theta) = \langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T}\rangle$, for an initial state $\theta \in \Theta$, time bound T, stepwise error bound $\epsilon > 0$, and time step $\delta > 0$, a $(\theta, T, \epsilon, \delta)$-simulation trace is a finite sequence $\beta = (\mathbf{v}_0, t_0), (\mathbf{v}_1, t_1), \ldots (\mathbf{v}_k, t_k)$, where $\mathbf{v}_0 = \theta, t_0 = 0, t_k = T$, and for each $i \in [k]$*

(i) $t_i = i\delta$,

(ii) $\forall \xi \in \mathcal{T}$ *such that* $\xi.\mathsf{fstate} = \mathbf{v}_{i-1}$ *and* $\xi.dur \geq \delta$, $|\xi(\delta).X - \mathbf{v}_i.X| \leq \epsilon$, *and*

(iii) $\forall \alpha \in \mathsf{Execs}_{\mathcal{A}}$ *such that* $\alpha.\mathsf{fstate} = \theta$ *and* $\alpha.dur \geq t_i$, $\alpha(t_i).loc = \mathbf{v}_i.loc.$

A $(\theta, T, \epsilon, \delta)$-simulation trace β is a sequence of samples starts at θ with duration T. The continuous state $\mathbf{v}_i.X$ is within ϵ distance to any trajectory starts at $\mathbf{v}_{i-1}$ with duration δ, if no transition can occur in time interval $[t_{i-1}, t_i]$. Moreover the sample points $\mathbf{v}_i$ have the same discrete state as any execution starts at θ evaluated at time t_i. Most simulation oracle guarantees that by letting sample period $\delta \to 0$, the stepwise error goes to zero $\epsilon \to 0$. Moreover, ϵ is of a higher order than δ, that is, $\epsilon \sim o(\delta)$.

2.6 Summary

In this chapter, we described the language for specifying a general class hybrid automata (HAs). We introduced simulation traces for HA. In Chapter 3, we present a simulation-based verification algorithm for a class of HA that is specified with differential equations. In Chapter 4, the presented algorithm is used to verify several examples specified in HA language, such as navigation benchmark, room heating system, satellite system and engine control system. In Chapter 5, we describe a special class of HA with differential inclusions and introduce a verification algorithm uses the algorithm presented for HA specified with differential inclusions.

CHAPTER 3

HYBRID SYSTEM WITH DIFFERENTIAL EQUATIONS

In this chapter, we present a simulation based algorithm to verify a class of hybrid automata (HA) where the flow condition is a set of differential equations, as shown in Equation (2.2). In Section 3.1, we define a class of deterministic HA. In Section 3.3, we introduce a sound algorithm to compute overapproximations of the reach sets of such HA. This algorithm first generates simulation traces start from different initial states. Then around each simulation trace, a tube is computed which guarantees to contain a set of executions start from a state close to the initial state of the simulation. In Section 3.4, we prove that the algorithm is complete in the sense it can compute the overapprixmation upto arbitrary accuracy.

3.1 System Properties

In this chapter we assume that the flow condition of each location of HA $\mathcal{A}$ is specified by differential equations, as shown in Equation (2.2). We will assume that the discrete transitions of the HA are deterministic. For the definition of HA given in Definition 1 this is accomplished by restricting the invariants to be disjoint and the transition guards to be contained within the boundary of the invariants.

Assumption 1 (Deterministic transition). *For HA $\mathcal{A} = \langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T} \rangle$, we assume*

(i) $\cup_{l \in \mathcal{L}} Inv(l)^c = Val(X)$,

(ii) $Inv(l) \cap Inv(l') = \emptyset$ *for any* $l, l' \in \mathcal{L}$, *and*

(iii) $Grd(l, l') \subseteq \partial Inv(l) \cap \partial Inv(l')$ *for any* $l, l' \in \mathcal{L}$.

In above assumption, (ii) together with (iii) guarantees that the invariants and guards are disjoint. As a result, from any state $\mathbf{x} \in Val(X)$, there at most exist one transition or a trajectory of positive duration. The first part of the above assumption is that the closures of the invariants cover the valuation of continuous variable.

Assumption 2 (Dwell time)**.** *The automaton has a minimum dwell time $\delta > 0$ such that along any execution, no two discrete transitions occur within δ time of each other.*

The concept of a dwell time was introduced in [35] to characterize the speed of mode switches in the context of analyzing stability of switched systems. Assumption 2 is satisfied by most well-designed systems that do not have Zeno executions.

Assumption 3 (Lipschitz)**.** *For each location $l \in \mathcal{L}$, the function f_l (the right hand side of the differential equation) is Lipschitz continuous with a Lipschitz constant $L_l > 0$.*

Assumption 4 (Bounded derivative difference)**.** *For $l, l' \in \mathcal{L}$, there exists an upper bound D such that for all $\mathbf{x} \in Grd(l, l')$, $|f_l(\mathbf{x}) - f_{l'}(\mathbf{x})| \leq D$.*

Assumption 3 is satisfied if f_i has a bounded derivative over $Inv(i)$. This assumption is commonly used to guarantee that, for any valuation $\mathbf{x} \in Inv(l)^o$ in the interior of any invariant, evolving according to Equation (2.2), there always exists a unique trajectory ξ with $\xi(0).X \in Inv(l)$ and $\xi.dur > 0$. If the guards are bounded then Assumption 3 implies Assumption 4.

3.2 Representative Cover

Consider any two executions $\alpha, \alpha' \in \mathsf{Execs}_{\mathcal{A}}$ which visit the same sequence of locations. From Assumption 3 and 4, we can infer that $\alpha(t)$ and $\alpha'(t)$ for some t can be arbitrary close if (i) the initial states of α and α' are close enough, and (ii) the times at which the same transitions occur on α and α' are close enough. Roughly, our algorithm works by constructing a tube that overapproximates the reach set of all executions starting from a set of initial states. For completeness, we will require that the overapproximations can be made tighter with appropriate choice of (smaller) initial sets. However, we make no assumption on the size of the initial set Θ of $\mathcal{A}$. Thus, a single simulation trace from a particular initial state is insufficient for tight reachability computation of the HA.

We define a type of cover for the initial set that groups together executions that visit the same sequence of locations upto a given time horizon. The executions start from a cover of the initial set can be represented and therefore approximated by a single simulation trace. First, we define the notion of an execution being represented by a initial state.

Definition 3 (Execution representation). *Fix any sample period $\delta > 0$, radius $\gamma > 0$ and a number k. An execution $\alpha \in \mathsf{Execs}_{\mathcal{A}}$ is (δ, γ, k)*-represented *by a $\theta \in \Theta$ if*

(i) $\alpha(0).\mathsf{fstate} \in B_\gamma(\theta)$*, and*

(ii) for α' with $\alpha'.\mathsf{fstate} = \theta$, for any $i = 0, 1, \ldots, k$, $\alpha(i\delta).loc = \alpha'(i\delta).loc$.

That is, an execution α is (δ, γ, k)-represented by θ if (i) it starts from a state in the γ ball of θ, and (ii) it visits the same sequence of location at each sample time as the execution starts from θ. We define $\mathcal{E}(\theta) \subseteq \mathsf{Execs}_{\mathcal{A}}$ be the set of all (δ, γ, k)-represented executions starting from θ. Roughly, any execution $\alpha \in \mathcal{E}(\theta)$ is close to the execution α' starts from θ in two ways. First, the initial states of α and α' can be at most γ away. Second, the difference in time when the same transition occurs on α and α' can be at most δ.

Proposition 1 (Representative covers for initial sets). *Fix an HA $\mathcal{A}$. For any sample period $\delta > 0$, a radius $\gamma > 0$ and k there, There exists a finite γ-cover of the initial state $C_\gamma = \{\theta_j\}_{j=1}^N$, called the (δ, γ, k)*-representative cover *of Θ, such that $\forall \alpha \in \mathsf{Execs}_{\mathcal{A}}$ with $\alpha.dur \geq k\delta$, $\exists \theta_j \in C_\gamma$ such that $\alpha \in \mathcal{E}(\theta_j)$.*

Proof. We will construct a such γ-cover and then prove it has the desired property. Fix any sample period $\delta > 0$, radius $\gamma > 0$ and k. Let $\mathsf{Execs}_{\mathcal{A}}(k\delta) \subseteq \mathsf{Execs}_{\mathcal{A}}$ be the subset of executions with duration no less than $k\delta$. First, we define an equivalence relation $(\sim)$ over the set $\mathsf{Execs}_{\mathcal{A}}(k\delta)$. For any $\alpha, \alpha' \in \mathsf{Execs}_{\mathcal{A}}(k\delta)$, $\alpha \sim \alpha'$ if for any $i = 0, 1, \ldots, k$, $\alpha(i\delta).loc = \alpha'(i\delta).loc$. That is, α and α' visit the same sequence of locations at each time $t = i\delta$. From Definition 1 of hybrid automaton, for any execution α and for each

$i = 0, 1, \ldots, k$, the location $\alpha(i\delta).loc$ is an element in the set $\mathcal{L}$. Thus the sequence of location, $(\alpha(0).loc, \alpha(\delta).loc, \ldots, \alpha(k\delta).loc)$, is an element in the set $\mathcal{L}^{k+1}$. Because $\mathcal{L}$ is finite, so is $\mathcal{L}^{k+1}$. Each equivalence class defined above corresponds to a unique element in $\mathcal{L}^{k+1}$. Thus, there can be at most $|\mathcal{L}|^{k+1}$ number of equivalence classes. Since Θ is bounded, the set of initial state of executions in any of the equivalence classes of $\sim$ is also bounded. Therefore, each such set has a finite γ-cover. The union of all the γ- covers of the equivalent classes is a (δ, γ, k)-representative cover of Θ. Thus the proposition follows. $\square$

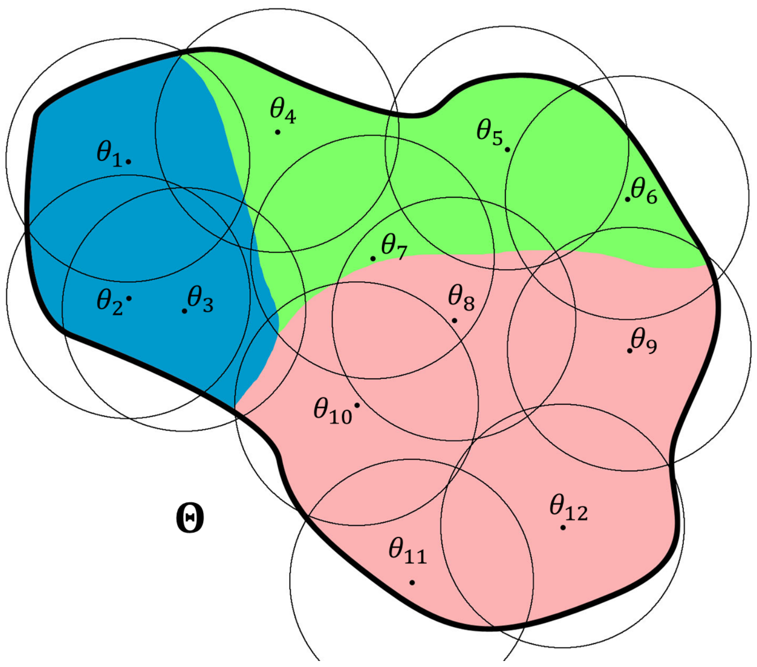


Figure 3.1: A (δ, γ, k)-representative cover of the initial set Θ

The proposition implies that there is always a (δ, γ, k)-representative cover of a bounded initial set, see an example in Fig 3.1. In this example with $k = 5$, let us say that the sequence of locations that executions staring from states in the green zone visit is $(1, 1, 1, 2, 2)$. The sequence of locations for the pink zone is $(1, 1, 2, 3, 3)$ and for the blue zone $(1, 1, 1, 1, 1)$. A $C_\gamma = \{\theta_j\}_{i=1}^{12}$ is constructed as a (δ, γ, k)-representative cover. Throughout the chapter we will assume that the (δ, γ, k)-representative covers can be computed.

Fix a sample period δ, a radius γ and sample number k. Fix a (δ, γ, k)-representative cover $C_\gamma = \{\theta_{i=1}\}^N$. We denote E_j be the set of states that are reachable with in time bound $T \triangleq k\delta$ by any execution $\alpha \in \mathcal{E}(\theta_j)$. It follows that $\mathsf{Reach}_{\mathcal{A}}(0,T) = \cup_{j=1}^{N} E_j$.

In the next section, we discuss how to compute an overapproximation of the set E_j based on the simulation trace starts from θ_j. Using these techniques, an overapproximation of $\mathsf{Reach}_{\mathcal{A}}(0,T)$ can be obtained by the union of the overapproximations of each E_j.

3.3 Overapproximation of Reach Set

Consider a HA $\mathcal{A}$ satisfying assumptions 1-4. Fix a sample period δ, a γ and a number k. Let C_γ be a (δ,γ,k)-representative cover. We defined a collection of set of execution $\{\mathcal{E}(j)\}_{j=1}^{N}$ such that each $\mathcal{E}(j)$ contains all executions starts in the ball $B_\gamma(\theta_j)$ and visit the same sequence of location as the execution starts at θ_j. We defined set E_j as the reachable set of executions in $\mathcal{E}(j)$. In this section, we introduce a simulation-based algorithm that computes an overapproximation of the set E_j.

Our approach is based on a simulation traces of the HA starting from θ_j of the type defined in Definition 2. Given simulation trace β, the computation involves two steps: first, we find the accumulated simulation error corresponding to every step, and then we propagate a tube around each sample point in β that is guaranteed to contain all executions $\alpha \in \mathcal{E}(j)$.

3.3.1 Estimating Accumulated Error

In Definition 2, we define a $(\theta,T,\epsilon,\delta)$-simulation trace $\beta = (\mathbf{v}_0,t_0),\ldots(\mathbf{v}_k,t_k)$ as a trace starts from θ with a step-wise error ϵ. We discuss how fast the error possibly accumulates. We fix the initial state $\theta \triangleq \theta_j \in C_\gamma$ as any state in the (δ,γ,k)-representative cover and the sampled trace β from θ of length $k+1$. Let $\{\gamma_i\}_0^k$ be a collection of error bounds for β that satisfy the following: for any execution $\alpha \in \mathcal{E}(j)$, for any $i \in \{0,\ldots,k\}$,

$$|\alpha(t_i).X - \mathbf{v}_i.X| \le \gamma_i. \tag{3.1}$$

That is, the state of any execution α in $\mathcal{E}(j)$, at time t_i, is within γ_i distance of the i^{th} sampled point $\mathbf{v}_i.X$. It follows Definition 3 that for any $\alpha \in \mathcal{E}(j)$,

$|\alpha(0).X - \theta_j| \leq \gamma$. That is, $\gamma_0 = \gamma$. We fix any $\alpha \in \mathcal{E}(j)$ in the rest part of this section. Using the following Lemma 2 and 4, the $\{\gamma_i\}_{i=0}^{k}$ can be computed inductively starting from $\gamma_0 = \gamma$.

Lemma 2. *If* $\forall t \in [t_i, t_{i+1}]$, $\alpha(t).loc = l$ *for some* $l \in \mathcal{L}$, *then*

$$\gamma_{i+1} \leq \gamma_i e^{L_l \delta} + \epsilon.$$

Proof. Fix i. Let $\delta \triangleq t_{i+1} - t_i$ be the gap in time between two samples. $\alpha(t).loc$ is a constant for $t \in [t_i, t_{i+1}]$. From Section 2.4, α is a concatenation of trajectories. From Definition 1, the set of trajectories is closed under prefix and suffix. Thus, there must be a trajectory $\xi \in \mathcal{T}$ such that $\alpha(t + t_i) = \xi(t)$ for all $t \in [0, \delta]$. Fix any $\xi' \in \mathcal{T}$ such that $\xi'(0).X = \mathbf{x}_i$ and $\xi'.dur \geq \delta$.

$$\begin{aligned} \gamma_{i+1} &= |\mathbf{v}_{i+1}.X - \alpha(t_{i+1}).X| = |\mathbf{v}_{i+1}.X - \xi(\delta).X| \\ &\leq |\mathbf{v}_{i+1}.X - \xi'(\delta).X| + |\xi'(t_{i+1}).X - \xi(\delta).X| \end{aligned} \tag{3.2}$$

From Definition 2, $|\mathbf{v}_{i+1}.X - \xi'(t).X| \leq \epsilon$. From Equation (2.2), the other term can be written in integral form.

$$|\xi'(\delta).X - \xi(\delta).X| = |\mathbf{v}_i.X + \int_0^\delta f_l(\xi'(t).X)dt - \xi(0).X - \int_0^\delta f_l(\xi(t).X)dt|$$

By triangular inequality, we have:

$$|\xi'(\delta).X - \xi(\delta).X| \leq |\mathbf{v}_i.X - \xi(0).X| + \int_0^\delta |f_l(\xi'(t).X) - f_l(\xi(t).X)|dt$$

By definition, $|\mathbf{v}_i.X - \xi(0).X| \leq \gamma_i$. From the condition of f_l, we have

$$|\xi'(\delta).X - \xi(\delta).X| \leq \gamma_i + \int_0^\delta L_l|\xi'(t).X) - \xi(t).X)|dt \tag{3.3}$$

From the Gronwall-Bellman inequality [36],

$$|\xi'(\delta).X - \xi(\delta).X| \leq \gamma_i e^{L_l \delta}$$

Combined with Equation (3.2), we get $\gamma_{i+1} \leq \gamma_i e^{L_l \delta} + \epsilon$. □

In the special case of a linear autonomous hybrid system we have the following corollary.

Corollary 3. *If* $\forall t \in [t_k, t_{k+1}]$, $\alpha(t).loc = l$ *and the differential that equation continuous states evolve follows has the form* $f_l(X) = A_l X + B_l$, *where* A_l *and* B_l *are constant matrices, then*

$$\gamma_{i+1} \leq \gamma_i |e^{A_l \delta}| + \epsilon.$$

On the other hand, if a transition occurs in the interval $[t_i, t_{i+1}]$, the following error bound holds.

Lemma 4. *If there exists* $\eta \in [0, t_{i+1} - t_i]$ *such that a single transition from location* l *to* l' *occurs in* α *at time* $t_i + \eta$, *then there exists a constant* $M > 0$ *such that*

$$\gamma_{i+1} \leq (\gamma_i + M/L_{l'})e^{L\delta} - M/L_j + \epsilon,$$

where $L = max\{L_l, L_{l'}\}$.

Proof. Fix i. Denote $\delta \triangleq t_{i+1} - t_i$. Fix an execution fragment $\alpha' \in \mathsf{Frags}_{\mathcal{A}}$ such that $\alpha'(0) = \mathbf{v}_i$ and $\alpha'.dur \geq \delta$.

Construct an auxiliary trace $\tau : [0, \delta] \mapsto Q$ with following properties: (i) $\tau(0) = \mathbf{v}_i$; (ii) $\forall t \in [0, t_i + \eta)$, $\frac{d}{dt}\tau.X = f_l(\tau.X)$; and (iii) $\forall t \in [t_i + \eta, t_{i+1})$, $\frac{d}{dt}\tau.X = f_{l'}(\tau.X)$. The execution fragment α' is a continuous mapping over $[0, \delta]$, thus its range is bounded. Denote R_k to be the reach set of continuous variables of α' over $[0, \delta]$. From Assumption 3 and 4, we can show that $|f_l(X) - f_{l'}(X)|$ over R_k is bounded. Denote $M = \sup_{X \in R_k} |f_l(X) - f_{l'}(X)|$.

By applying triangular inequality we have:

$$|\mathbf{v}_{i+1}.X - \alpha(t_{i+1}).X| \leq |\mathbf{v}_{i+1}.X - \alpha'(\delta).X| + |\alpha'(\delta).X - \tau(\delta).X| + |\tau(\delta).X - \alpha(t_{k+1}).X| \tag{3.4}$$

The continuous states of τ and that of α evolve follow identical differential equation for the whole interval $[t_i, t_{i+1}]$. Thus $|\mathbf{v}_{i+1}.X - \alpha'(\delta).X| + |\tau(\delta).X - \alpha(t_{k+1}).X|$ is analog to the right hand side of Equation (3.2). Following the same steps as in Lemma 2, we have

$$|\mathbf{v}_{i+1}.X - \alpha'(\delta).X| + |\tau(\delta).X - \alpha(t_{i+1}).X| \leq \gamma_i e^{L\delta} + \epsilon.$$

Note that α' and τ have the same first state $\mathbf{v}_i$. Thus the difference between

$\alpha'(t)$ and $\tau(t)$ comes from the different times the transitions occur. Thus:

$$\begin{aligned} & |\alpha'(\delta).X - \tau(\delta).X| \leq \textstyle\int_0^\delta |f_l(\alpha'(t).X) - f_{l'}(\tau(t)).X|dt \\ \leq \;& \textstyle\int_0^\delta |f_l(\alpha'(t).X) - f_{l'}(\alpha'(t).X)|dt + \int_0^\delta |f_{l'}(\alpha'(t).X) - f_{l'}(\tau(t).X)|dt \\ \leq \;& \delta M + \textstyle\int_0^\delta L_{l'}|\alpha'(t).X - \tau(t).X|dt \end{aligned}$$

From Gronwall-Bellman inequality,

$$|\alpha'(\delta).X - \tau(\delta).X| \leq (M/L_{l'})(e^{L_{l'}\delta} - 1).$$

Combining the above equation with Equation (3.4), from the definition of γ_{i+1}, we have $\epsilon_{i+1} \leq (\gamma_i + M/L_{l'})e^{L\delta} - M/L_{l'} + \epsilon$. Thus the lemma follows. □

For bounded invariants $Inv(l)$ and $Inv(l')$, setting $M = \sup_{X \in Inv(l) \cup Inv(l')} |f_l(X) - f_{l'}(X)|$ satisfies the above condition and makes the proof to go through, but a smaller range for X based on reachability analysis from $\mathbf{v}_i$ can provide tighter bounds.

```
1  $\gamma_0 \leftarrow \gamma$;
2  for $i = 0 : k-1$ do
3      $l \leftarrow \mathbf{v}_i.loc$;  $l' \leftarrow \mathbf{v}_{i+1}.loc$;
4      if $l = l'$ then
5          $\gamma_{i+1} \leftarrow \gamma_i e^{L_l \delta} + \epsilon$;
6      end
7      else
8          $M \leftarrow \sup_{X \in Inv(l) \cup Inv(l')} |f_l(X) - f_{l'}(X)|$;
9          $\gamma_{i+1} \leftarrow (\gamma_i + M/L_{l'})e^{L\delta} - M/L_{l'} + \epsilon$;
10     end
11 end
12 return $\{\gamma_i\}_{i=0}^k$;
```

Algorithm 1: Algorithm to compute $\{\gamma_i\}_{i=0}^k$

An algorithm that computes $\{\gamma_i\}_{i=0}^k$ is presented in Algorithm 1. Starting from $i = 0$, the algorithm checks whether a transition takes place in $[t_i, t_{i+1}]$ (line 4). If no transition takes place, γ_{i+1} is computed using Lemma 2 (line 5). Otherwise, it is computed using Lemma 4 (line 8-9). Recall that the sample trace β starts from some θ_j and $\mathcal{E}(j)$ denotes the set of executions start in the ball $B_\gamma(\theta_j)$ and visit the same sequence of location as the execution starts at

θ_j. It follows Lemma 2 and 4 that, for any $\alpha \in \mathcal{E}(j)$ and any $i = 0, 1, \ldots, k$, $|\alpha(t_i).X - \mathbf{v}_i.X| \leq \gamma_i$, with γ_i computed by Algorithm 1.

3.3.2 Reach Set Between Sampled Points

We presented an Algorithm (Algorithm 1) that computes the sequence of accumulated error ($\{\gamma_i\}_{i=0}^k$) of a given $(\theta, T, \delta, \epsilon)$-simulation trace ($\beta$). In this section, we construct tubes between each consecutive samples. We will show later that under our construction, the union of such tubes contains the set of reachable state by executions starts from a state in a ball $B_\gamma(\theta)$ corresponds to β. This tube is the union of all tube segments between every two consecutive sample points. To build the tube segments, we execute Algorithm 2.

```
1 $\sigma \leftarrow \gamma_i$;
2 do
3     $\sigma \leftarrow b * \sigma$;
4     $B \leftarrow B_\sigma(\mathbf{v}_i.X)$;
5     $m \leftarrow \sup_{\mathbf{x} \in B} |f(\mathbf{x})|$;
6 while $\sigma - m\delta < \gamma_i$;
7 $f_{min} \leftarrow \inf_{X \in B} f(X)$;  $f_{max} \leftarrow \sup_{X \in B} f(X)$;
8 $R_i \leftarrow Post(B_{\gamma_i}(\mathbf{v}_i.X), \delta, f_{min}, f_{max})$;
9 return $R_i$;
```

Algorithm 2: Algorithm to compute R_i

The reachable tube is overapproximated by a convex set R_i. The do-while loop (lines 2-6) iteratively computes a ball B centered at $\mathbf{v}_i.X$ with the radius σ which contains the bounded reach set from $\mathbf{v}_i$ for δ time. The estimate of σ is bloated by a constant factor $b > 1$ in each iteration, until σ is grater than γ_i added to the bound of change in the norm of continuous state in a δ interval ($m * \delta$). Given the ball B, the following part of the algorithm (lines 6-8) computes the set $R_i \subseteq B$. The vector f_{min} (f_{max}) is the element-wise lower (upper) bound of the derivative of $\alpha.X$. The function $Post(B_{\gamma_i}(\mathbf{v}_i.X), \delta, f_{min}, f_{max})$ returns the the set of states can be reached from the ball $B_{\gamma_i}(\mathbf{v}_i.X)$ up to δ time with derivative bounded by f_{min} and f_{max}.

Let the simulation trace β starts from a state θ_j in a (δ, γ, k)-representative cover. For any $\alpha \in \mathcal{E}(j)$, we claim that R_i contains $\alpha(t)$ with $t \in [t_k, t_{k+1}]$. First, we prove the following lemma based on the assumption on termination of Algorithm 2.

Lemma 5. *Assume that the while loop in Algorithm 2 terminates. For all $\alpha \in \mathcal{E}(j)$, for all $t \in [t_i, t_{i+1}]$, $\alpha(t) \in B$, where B is computed by the while loop.*

Proof. Fix any $\alpha \in \mathcal{E}(j)$ and any i. In Section 3.3.1 we show that $\alpha(t_i).X \in B_{\gamma_i}(\mathbf{x}_i)$ (Equation (3.1)). We will prove the lemma by contradiction. Assume on the contrary, $\alpha(t)$ with $t \in [t_k, t_{k+1}]$ is not contained by B. Note that $\alpha(t_i) \in B_{\gamma_i}(\mathbf{v}_i)$. The radius of B is σ which is strictly greater than γ_i (line 1, 3). So we have $\alpha(t_i) \in B$. Thus, there must be a first time t^* strictly before t_{i+1} such that α reaches the boundary of B. That is, there exist a time $t^* \in [t_i, t_{i+1})$ such that (i) $|\alpha(t^*).X - \mathbf{v}_i.X| = \sigma$, and (ii) $\forall t \in [t_i, t^*]$, $\alpha(t) \in B$.

From triangular inequality,

$$|\alpha(t^*).X - \mathbf{v}_i.X| \leq |\alpha(t_i).X - \mathbf{v}_i.X| + |\alpha(t_i).X - \alpha(t^*).X|. \tag{3.5}$$

By definition $|\alpha(t_i).X - \mathbf{v}_i.X| \leq \gamma_i$. From the integral form of the flow condition, $|\alpha(t_i).X - \alpha(t^*).X| = |\int_{t_i}^{t^*} f(\alpha(t).X)dt|$. We know that for $t \in [t_i, t^*]$, $\alpha(t) \in B$. By definition of m in Algorithm 2, $|f(\alpha(t).X)| \leq m$. Thus, $|\int_{t_i}^{t^*} f(\alpha(t).X)dt \leq |t^* - t_i|m < \delta m$. The strict inequality is derived from the fact t^* is strictly before t_{i+1}. Substitute the above inequalities into Equation 3.5 we get

$$|\alpha(t^*).X - \mathbf{v}_i.X| < \gamma_i + m\delta. \tag{3.6}$$

Recall that we choose t^* such that $|\alpha(t^*).X - \mathbf{v}_i.X| = \sigma$. Combined with Equation (3.6), we derived that $\sigma < \epsilon_i + m\delta$, which contradicts to the terminating condition. Then the lemma follows. □

Let $L = \max_{l \in \mathcal{L}} L_l$ denote the largest Lipschitz constant. We will show that Algorithm 2 terminates and is sound.

Lemma 6. *The while loop of Algorithm 2 terminates if $L\delta < 1$. Moreover, for any $\alpha(t) \in \mathcal{E}(j)$ and for any $t \in [t_i, t_{i+1}]$, $\alpha(t).X \in R_i$, with R_i computed by the algorithm.*

Proof. First we prove termination. Fix any i. We will use the superscript (j) to denote variable values after the j^{th} iteration. Let $V^{(j)} \triangleq \sigma^{(j)} - m^{(j)}\delta$. We show that there exists a constant $c > 0$ such that for each j, $V^{(j+1)} > V^{(j)} + c$, and thus, eventually $V^{(j)} > \gamma_i$ which terminates the while loop.

$$\begin{aligned} V^{(j+1)} - V^{(j)} &= b^{j+1}\gamma_i - m^{(j+1)}\delta - b^j\gamma_i - m^{(j)}\delta \\ &= b^j(b-1)\gamma_i - \delta[m^{(j+1)} - m^{(j)}] \\ &= b^j(b-1)\gamma_i - \delta|f(X^*)| + \delta m^{(j)}. \end{aligned} \tag{3.7}$$

where $X^* \in B_{\sigma^{(j+1)}}(\mathbf{v}_i.X)$ is the state at which $|f(X^*)| = m^{(j+1)}$ is realized. Next, we estimate a lower bound on $m^{(j)}$ in terms of $f(X^*)$. From the definition of L, for all $X \in B^{(j)}$, we have $|f(X^*)| - |f(X)| \leq |f(X^*) - f(X)| \leq L|X^* - X|$. Thus, $m^{(j)} \geq |f(X)| \geq |f(X^*)| - L|X^* - X|$. In particular, if we choose the $X \in B_{\sigma^{(j)}}(\mathbf{v}_i.X)$ that is closest to X^*, then we get the tightest bound $m^{(j)} \geq |f(X^*)| - Lb^j(b-1)\gamma_i$. Replacing this in Equation (3.7) and simplifying, we get $V^{(j+1)} - V^{(j)} > c = (1 - L\delta)(b-1)\gamma_i$. Provided $L\delta < 1$, we have $c > 0$. It follows that there exists a finite j such that $V^{(j)} > \gamma_i$, which terminates the loop.

Lemma 5 suggests that B is an overapproximation for the reach set of α. The algorithm computes a better overapproximation using B. The f_{min} and f_{max} computed by the algorithm satisfies $f_{min} \leq f(\alpha(t).X) \leq f_{max}$ for each $t \in [t_i, t_{i+1}]$. Since, $\alpha(t_i) \in B_{\gamma_i}(\mathbf{v}_i.X)$ it follows that the overapproximation computed by $Post(B_{\gamma_i}(\mathbf{v}_i.X), \delta, f_{min}, f_{max}) = R$ contains $\alpha(t)$ for all $t \in [t_i, t_{i+1}]$. □

3.3.3 Overapproximation of Bounded Reach Set

In Lemma 6, we prove that Algorithm 2 computes a R_i contains the reach set of executions in $\mathcal{E}(j)$ in time interval $[t_i, t_{i+1}]$. We introduce Algorithm 3 for computing an overapproximation of the reach set $\mathsf{Reach}_{\mathcal{A}}(0, T)$. We assume that T is divisible by δ ($T = k\delta$). In Line 2 of the algorithm, the initial set of $\mathcal{A}$ is partitioned into a (δ, γ, k)-representative cover with N elements. For the center point of each part, Line 4 computes a $(\theta_j, T, \epsilon, \delta)$-simulation trace using some oracle. Line 5 computes a sequence of error bounds using Algorithm 1. And Line 7 computes the reach set in time interval $[t_i, t_{i+1}]$ using Algorithm 2.

```
$R \leftarrow \emptyset$;
$\{\theta_j\}_{j=1}^{N} \leftarrow Partition(\delta, \gamma, k)$;
for $j = 1 : N$ do
    $\beta \leftarrow Simulation(\theta_j, T, \epsilon, \delta)$;
    $\{\gamma_i\}_{i=0}^{k} \leftarrow errorbnd(\gamma, \delta, \beta)$;
    for $i = 0 : k - 1$ do
        $R_i \leftarrow reach(\gamma_i, \beta)$;
        $R \leftarrow R \cup R_i$;
    end
end
return $R$;
```

Algorithm 3: Main Algorithm to compute $\mathsf{Reach}_{\mathcal{A}}(0, T)$

Then the union of all computed R_is is an overapproximation of the reach set of $\mathcal{A}$.

We will prove our first main theorem on the soundness of Algorithm 3 based on the lemmas stated previously.

Theorem 1. *Let R be the output of Algorithm 3, then $R \supseteq \mathsf{Reach}_{\mathcal{A}}(0, T)$.*

Proof. Fix any execution $\alpha \in \mathsf{Execs}_{\mathcal{A}}$. Let $\{\theta_j\}_{j=1}^{N}$ (line 2) be a (δ, γ, k)-representative cover of the initial state. From Proposition 1, there exists a $j \in \{1, \ldots, N\}$ such that $\alpha \in \mathcal{E}(j)$. That is, (i) $\alpha.\mathsf{fstate} \in B_{\gamma}(\theta_j)$, and (ii) for $\alpha'.\mathsf{fstate} = \theta_j$ and $i = 0, 1, \ldots, k$,

$$\alpha(i\delta).loc = \alpha'(i\delta).loc. \tag{3.8}$$

Let $\beta = (\mathbf{v}_0, 0), \ldots, (\mathbf{v}_k, k\delta)$ be a $(\theta_j, T, \epsilon, \delta)$-simulation trace computed in line 4. From Definition 2, $\mathbf{v}_i.loc = \alpha'(i\delta).loc$. For each $i = 0, 1, \ldots, k$. Combined with Equation (3.8) we have $\alpha(i\delta).loc = \mathbf{v}_i.loc$. That is,evaluated at each sample time $i\delta$, the execution α has the same location the sample trace β. Let R_i be computed from line 7 for some i. Form Lemma 6, we have for any $t \in [t_i, t_{i+1}]$, $\alpha(t).X \in R_i$. We reapply the same analysis to each sample point $i = 0, \ldots, k$ and get $\alpha(t) \in \cup_{i=0}^{k} R_i$ for all $t \in [0, T]$. From line 8, we have $\cup_{i=0}^{k} R_i \subseteq R$. Thus we have $\alpha(t) \subseteq R$ for all $t \in [0, T]$. Recall that α is selected randomly in $\mathsf{Execs}_{\mathcal{A}}$. Thus $\mathsf{Reach}_{\mathcal{A}}(0, T) \subseteq R$ follows. □

3.4 Completeness of the Algorithm

In the previous section, we present an algorithm to compute an overapproximation of the bounded reach set of HA $\mathcal{A}$. In this section, we present that the overapproximation can be made arbitrarily tight given fine enough initial partition γ, sample period δ and stepwise simulation error ϵ.

First, let $\{\gamma_i\}_{i=0}^{k}$ be made using Algorithm 1, we prove the following lemma.

Lemma 7. *Let $\{\gamma_i\}_{i=0}^{k}$ be the output of Algorithm 1 with inputs $\gamma, \delta, \epsilon > 0$. We have $\gamma_i \to 0$ as $\gamma, \delta, \epsilon \to 0$ for all i.*

Proof. By definition γ_i is an increasing sequence. So it is sufficient to show that $\gamma_k \to 0$ as $\gamma, \delta, \epsilon \to 0$. From Assumption 2, there only finite many transitions can occur on any execution of $\mathcal{A}$ in bounded time T. Let $L > 0$ be the maximum Lipschitz constant. We prove this lemma using induction on the number of transitions. Assume that there are in total M transitions on the simulation trace β. For $1 \le m \le M$, let the m^{th} transition occurs in intervals $[t_i, t_{i+1}]$ with $i = a_m$. Denote $a_{M+1} - 1 \triangleq k$.

Base case: Before the first transition occurs, $0 \le i < a_1$. We have

$$\gamma_{a_1-1} \le \gamma_{a_1-2} e^{L\delta} + \epsilon.$$

Recursively apply the above inequality, we get

$$\gamma_{a_1-1} \le \gamma_0 e^{Lt_{a_1}} + \epsilon \frac{e^{Lt_{a_1}} - 1}{e^{L\delta} - 1} \le \gamma e^{LT} + \frac{\epsilon(e^{LT} - 1)}{L\delta}. \tag{3.9}$$

Recall in Section 2.5, we assume that ϵ is of a higher order of δ. Thus $\gamma_{a_1-1} \to 0$, as $\gamma, \delta, \epsilon \to 0$.

Induction: Assume that $\gamma_{a_m-1} \to 0$, as $\gamma, \delta, \epsilon \to 0$, we will prove that $\gamma_{a_{m+1}-1} \to 0$. From Line 7-10 of Algorithm 1, we have

$$\gamma_{a_m} \le \gamma_{a_m-1} e^{L\delta} + M/L\left(e^{L\delta} - 1\right) + \epsilon \le \gamma_{a_m-1} e^{L\delta} + M\delta + \epsilon.$$

It follows that if $\gamma, \delta, \epsilon \to 0$, $\gamma_{a_m} \to 0$. Using the same recursion as for Equation (3.9), we have

$$\gamma_{a_{m+1}-1} \le \gamma_{a_m} e^{LT} + \frac{\epsilon(e^{LT} - 1)}{L\delta}.$$

As $\gamma, \delta, \epsilon \to 0$, we proved that $\gamma_{a_m} \to 0$. Thus $\gamma_{a_{m+1}-1} \to 0$.

Using the induction, we have $\gamma_k = \gamma_{a_{M+1}-1} \to 0$ as $\gamma, \delta, \epsilon \to 0$. Then the lemma follows. □

For an HA $\mathcal{A}$ satisfies Assumption 1-4, we will show that we can compute the reach set of $\mathcal{A}$ with arbitrary accuracy using Algorithm 3.

Theorem 2. *For any accuracy $d > 0$ and time bound $T > 0$, there exists $\delta, \gamma, \epsilon > 0$ such that if R is computed by Algorithm 3 using δ, γ, ϵ, then*

$$R \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{A}}(0, T), d).$$

Proof. In line 8 of Algorithm 2, the set R_i is obtained by propagate the ball $B_{\gamma_i}(\mathbf{v}_i.X)$ for a δ interval of time using the maximum and minimum rate of evolution (f_{max}, f_{min}). Recall that f is Lipschitz continuous (Assumption 3), thus f_{maxi}, f_{min} is always bounded in bounded domain. Thus, for sufficiently small initial radii γ_i and period δ, the diameter of the set R_i can be made less than d. That is $D(R_i) < d$. In Lemma 7, we show that for sufficiently small $\gamma, \delta, \epsilon > 0$, γ_i can be made arbitrarily small. Thus, we can choose sufficiently small γ, δ, ϵ such that $D(R_i) < d$ for any R_i computed in line 7 of Algorithm 3.

Let $\{\theta_j\}_{j=1}^N$ be the (δ, γ, k)-representative cover. Let $R_i^{(j)}$ denote the set R_i computed in the j^{th} for loop (line 3-10) of Algorithm 3. Recall that we defined E_j be the set of state reachable by executions in $\mathcal{E}(j)$. By definition, we have $E_j \subseteq \mathsf{Reach}_{\mathcal{A}}(0, T)$ for each j. Recall that the diameter of $R_i^{(j)}$ is less than d. Thus $R_i^{(j)} \subseteq \mathsf{Expand}(E_j, d)$ for each i. Thus, $R = \cup_j \cup_i R_i^{(j)} \subseteq \cup_j E_j \subseteq \mathsf{Reach}_{\mathcal{A}}(0, T)$ follows. □

3.5 Summary

In this chapter, we presented a reachability algorithm for a class of HA specified with differential equations. We made Assumptions 1-4 on the systems' determinism, dwell time and bounds on dynamics. We introduce a (δ, γ, k)-representative cover of the initial set of the HA such that executions from the same cover have similar behaviors. For each cover, a simulation trace is generated and used to compute arbitrarily tight overapproximations of the reachable state of the executions from the cover. The algorithm involves two steps. First, the overapproximated reach set at each sample time is generated. Then, each reach set on the sample time is propagated for a sample period. We proved the soundness and completeness of the algorithm.

Our simulation-based approach is similar to the one presented in [37]. In that paper, different types of annotations, including Lipschitz constant as we used here, are used to compute the bound on distance between the reachable state on a trajectory and its simulation points. While our approach can handle transitions.

CHAPTER 4

EXPERIMENTAL EVALUATION

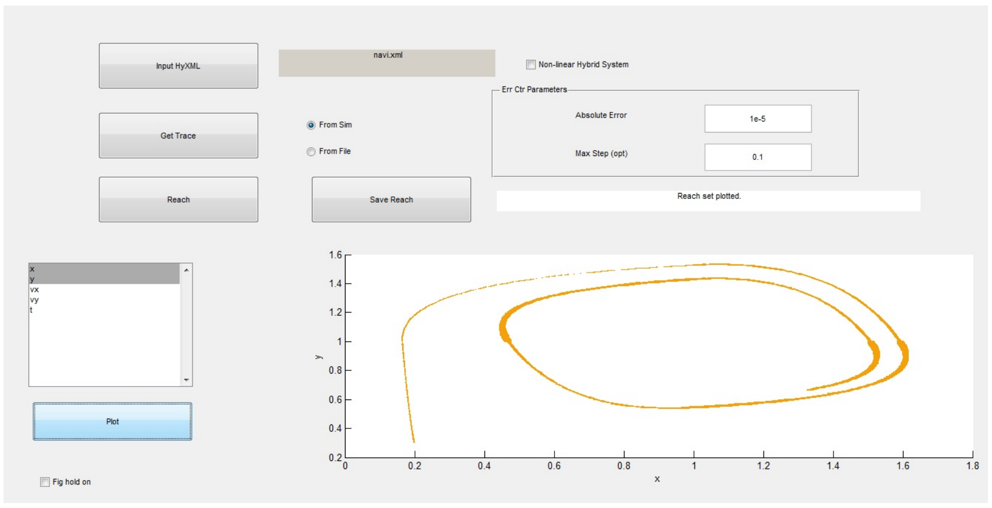
We present the experimental evaluation of the simulation-based verification algorithm with respect to running time and memory usage. Recall that the input to the algorithm is a simulation trace β with l states, an unsafe set U, and a model of the system. The algorithm iteratively computes over-approximations of the reach set of any execution that may correspond to β and decides if the ϵ-tube is safe or unsafe with respect to U. We have implemented the algorithm in a tool, the *Hybrid Trace Verifier (***HTV***)* [1]. The user interface and an experiment result is shown in Figure 4.1. **HTV** can handle both linear and non-linear hybrid automata models. This implementation uses MATLAB's Optimization Toolbox for maximizing $f(x)$ over compact sets. In this experiments, we consider small initial sets where no partition is needed. The experimental results reported here were performed on a Intel dual core 2.26GHz processor. In the following Table we show the performance of HTV in handling realistic benchmarks and scalability to higher dimensional models.

Table 4.1: Tool performance

Benchmark	#var,loc	rt (s)	#sp	ct (s)	#tra.	err.	m. (Kb)
Room Heating I	3,3	2	168	6.6	4	0.34	24.4
Room Heating II	10,10	2	168	22.6	4	0.67	70
Navigation I	4,4	4	192	9.6	3	0.04	36
Navigation II	4,9	5	232	14.0	4	0.06	43
Navigation III	4,16	7	316	17	5	0.12	58
Navigation IV	8,16	7	321	36	5	0.13	110
Satellites	2,2	3.5	564	14	1	0.15	58
Engine Control	4,2	4	200	11	4	2.50	37

In Table 4.1, the entries in the first column indicate the benchmarks veri-

[1] **HTV**and case study files are available from: `https://wiki.cites.uiuc.edu/wiki/display/MitraResearch/HTV`


navi.xml

Err Ctr Parameters

Absolute Error 1e-5

Max Step (opt) 0.1

[✓ From Sim] From File

Reach set plotted.

Figure 4.1: The user interface and an experiment result

fied. The following columns strand for (i) the number of continuous variables and the number of locations in this benchmark, (ii) the real time measured in seconds of the reachability computation, (iii) the number of simulation samples generated, (iv) the computation time for running the tool measured in seconds, (v) the number of transitions occur on the trajectory, (vi) the maximum error of the computation, and (vii) the memory used for the computation measured in Kb.

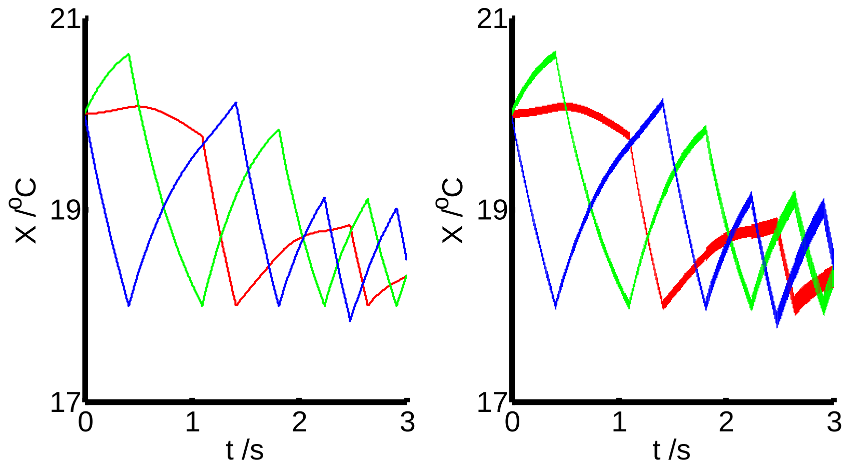


Figure 4.2: Room heating benchmark with 3 rooms

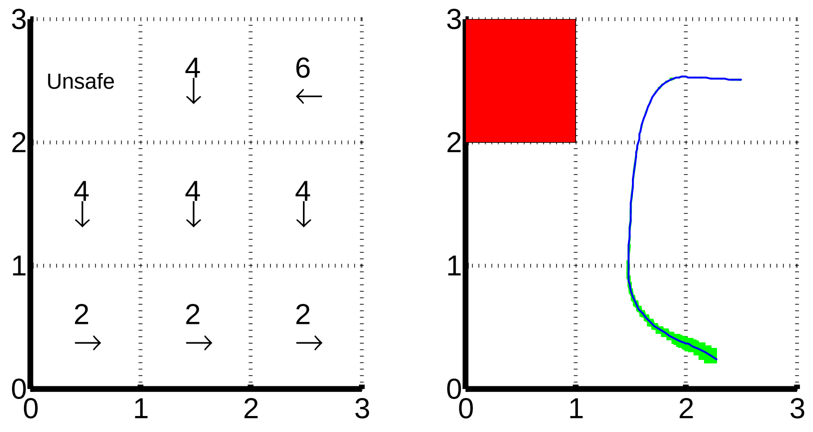


Figure 4.3: Overapproximation of reach set for the navigation system.

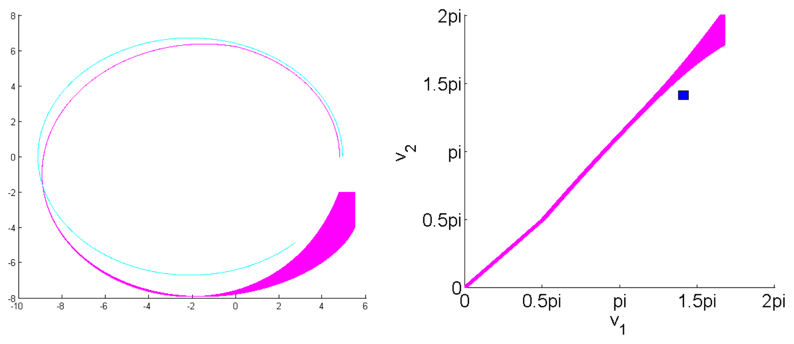


Figure 4.4: Overapproximation of reach set for the two satellites system.

4.1 Room Heating

The **Room heating** benchmark [38] (Figure 4.2) models a building with several rooms that are heated by heaters that can be turned off and on. The temperature dynamics is captured by linear differential equations. The discrete transitions capture heater control strategies. For example, heater in room a is turned off and the one in room b is turned on when (1) the temperature of room b is less than 18C, (2) a has the highest temperature among rooms adjacent to b, and (3) the temperature of room a is at least 1 degree higher than that of room b. An example of benchmark is specified

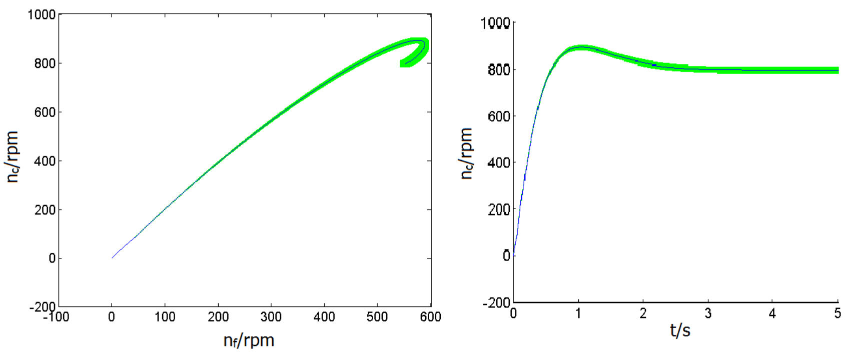


Figure 4.5: Overapproximation of reach set for the hybrid engine control system.

with the following differential equation:

$$\dot{x} = \begin{bmatrix} -0.9 & 0.5 & 0 \\ 0.5 & -1.3 & 0.5 \\ 0 & 0.5 & -0.9 \end{bmatrix} x + \begin{bmatrix} 0.4 \\ 0.3 \\ 0.4 \end{bmatrix} u + \begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 8 \end{bmatrix} h,$$

where $x \in \mathbb{R}^3$ is a vector of the temperatures in each room, $u \in \mathbb{R}$ is the outdoor temperature, and $h \in \{0,1\}^3$ is a vector of boolean variables, the i^{th} entry of which is set to 1 if the heater in the corresponding room is on.

The safety property of interest is that the temperature in all rooms remain above a threshold, say 17C degree. Higher dimensional models are created by increasing the number of rooms.

Figure 4.2 illustrates an instance of the benchmark with 3 rooms, whose temperature are plotted in red, green and blue respectively. The subfigure on the left shows an actual execution from initial state $[20, 20, 20]$ and heaters on in rooms 1 and 2. And the subfigure on the right illustrates an overapproximation of reach set computed by **HTV**. The overapproximation stays above 17C indicating safety.

4.2 Navigation

The **navigation** benchmark [38] (Figure 4.3) is a linear hybrid automaton which models positions of one or more particles in a partitioned plane.

Each partition is associated with a constant external force which acts on the particle which then moves according to Newton's laws. We consider a version of the 3×3 of this benchmark. This is a deterministic hybrid automaton with 4 continuous variables $X = (x, y, vx, vy)$ and 9 locations $\mathcal{L} = \{(i,j) | i,j \in \{1,2,3\}\}$. The system evolves according to differential equation $\dot{X} = AX - Bu(i,j)$, where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.2 & 0.1 \\ 0 & 0 & 0.1 & -1.2 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1.2 & 0.1 \\ 0.1 & -1.2 \end{bmatrix}.$$

The input u is assigned to be a constant in each mode

$$u(i,j) = [\sin((\pi c(i,j))/4), \cos((\pi c(i,j))/4)]^T,$$

where $c(i,j) \in \{0, 1, \cdots, 7\}$ determines a direction vector of location (i,j). The safety property of interest is that the particles never reach some partition simultaneously. An experiment result is shown in Figure 4.3. The subfigure on the left illustrate the inputs to each modes and the unsafe region. The subfigure on the right plot an experiment result where the red region is the unsafe set, the blue curve is the true execution and the green region is the result of reachability computation. Because the overapproximated reach set is disjoint from the unsafe region in this experiment, we prove safety.

4.3 Satellite

The nonlinear **satellite system** [39] models the angular position of a pair of satellites with nonlinear differential equations. The discrete states model the orbit that the satellites are following and the orbits change through thrusting. The safety property of interest is that the satellites never come closer than some threshold value.

The continuous state of the system $X = (\nu_1, \nu_2)$ captures the angular position of the two satellites. The discrete state $\mathcal{L} = \{1, 2\}$ captures the two possible orbits of the active satellite. The guards model the relative angular positions at which the orbital transition must occur. The differential equation for the passive satellite is

$$\dot{\nu_1} = \frac{1.0077}{(1 + 0.3\cos(\nu_1))^2}.$$

The differential equations for the the active satellite is

$$\dot{\nu_2} = \frac{0.9933}{1 + 0.3\cos(\nu_2)^2}$$

in orbit 1 and

$$\dot{\nu_2} = \frac{1.0446}{(1 + 0.25\cos(\nu_2 - 0.44))^2}$$

in orbit 2.

The computed reach set is shown in Figure 4.4. The figure on the left is the reach set in Cartesian coordinate. The pink area contains the execution of passive satellite and the cyan area contains the execution of active satellite. The figure on the right is the phase portrait. The darker tiny rectangle denotes the unsafe region near the intersection of the two orbits.

4.4 Engine Hybrid Control

The **Engine control system** models a switched linear jet engine control system. The controller switches between multiple adaptive control laws. The continuous variables are the fan speed, the core speed and two adaptive control parameters. The discrete state captures the control law. The required safety property is that the core speed remains below a threshold.

We consider an example with 4 continuous variables $X = (n_f, n_c, x_1, x_2)$ and 2 locations $\{1, 2\}$. The dynamics in each location is $\dot{X} = A_iX + u_i$, $i \in \{1, 2\}$. Where,

$$A_1 = \begin{bmatrix} -3.961 & 0.7344 & 672.7 & 0 \\ -3.704 & -1.774 & 1437 & 0 \\ -0.004285 & 0 & 0 & 0 \\ -0.01497 & 0.007887 & 5.543 & -5.425 \end{bmatrix}, u_1 = \begin{bmatrix} 1973 \\ 4257 \\ 2.354 \\ 11.92 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -2.145 & 0.4919 & 0 & 672.7 \\ 0.1758 & -4.394 & 0 & 1437 \\ -0.0301 & -0.02322 & -12.74 & 12.74 \\ 0 & -0.002217 & 0 & 0 \end{bmatrix}, u_2 = \begin{bmatrix} 699 \\ 1493 \\ -22.14 \\ 1.264 \end{bmatrix}$$

The invariants of locations are $Inv(1) : w \leq 0$, $Inv(2) : w > 0$. Where $w = -0.0027n_f + 0.001823n_c + x1 - x2 + 1.0468$. The required safety property is that n_c remains below a threshold.

The experiment result is illustrated in Figure 4.5. The subplot on the left is the phase portrait of the fan speed vs. the core speed. The subplot on the right illustrate the reach set of the core speed as a function of time.

In this case the A_i matrices are Hurwitz. In each location, different executions converge to each other. According to Corollary 3, the error bound shrinks when $\epsilon_k > \frac{c}{1-||e^{A(t_{k+1}-t_k)}||}$.

4.5 Discussion

For each case study, first, from a SLSF model a sampled trace is generated. Secondly, a hybrid automaton translation representing the SLSF model is obtained using the HyLink tool [10]. Performance of the tool is shown in Table 4.1. The second column gives the number of continuous variables and locations. The following columns give the duration of simulation trace (Real time), the number of sampled points (l), the run time of **HTV**for computing the overapproximation of reach set (Reach time), the number of transitions, the maximum error in $\{\epsilon_k\}$'s, and the memory used to store the reachable set.

We observe that **HTV**'s running time increases roughly linearly with the number of continuous variables of the system. Furthermore, the memory requirement for **HTV**grows only linearly with the number of sample points and the dimension of the system. This is because **HTV**stores the sampled trace and the polyhedral overapproximation of the reach set between a pair of consecutive sample points. These observations suggest that this combined static analysis and simulation based approach may scale to larger problems.

CHAPTER 5

DYNAMICAL SYSTEMS WITH DIFFERENTIAL INCLUSIONS

We will present an approximate reachability algorithm for hybrid automata with differential inclusions. The algorithm computes overapproximation of bounded time reach set and it is guaranteed to be complete. In Section 5.1, we formally define several concepts including dynamical system with symmetric uncertainty, shrink concretization and shift abstraction. Then in Section 5.2, we prove that the reach set of a dynamical system $\mathcal{A}$ can be upper and lower bounded by the reach set of its shrink concretization $\mathcal{B}$. In Section 5.3 we present a reachability algorithm for dynamical system with symmetric uncertainty $\mathcal{A}$. The algorithm runs iteratively over k time intervals each of which has length $\delta > 0$. In the $i^{(th)}$ iteration, we first define a shrink concretization of $\mathcal{A}$ shifted by $i\delta$ in time. The concretization we defined is a dynamical system specified by differential equations. Then the reach set of the shrink concretization in a time interval is computed using Algorithm 3 (in Chapter 3). Finally, the reach set of $\mathcal{A}$ is computed using the reach set of its shrink concretization. The majority of the existing reachability algorithms for system with differential inclusions work for rectangular dynamics [26, 27] or linear differential inclusions [29, 30]. To the best of our knowledge, this is the first algorithm for computing arbitrarily precise bounded time reach sets for nonlinear differential inclusions.

5.1 Preliminaries of Dynamical Systems

5.1.1 Dynamical System

In this section, we discuss dynamical systems which is a special class of HA, where no transitions can occur.

Definition 4 (Dynamical system)**.** *We assume that the HA $\mathcal{A} = \langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T}\rangle$ satisfies*

(i) the set of discrete variable $\mathcal{L} = \{true\}$ is a singleton,

(ii) the set of guard Grd is empty such that no transition can be enabled, and

(iii) the invariant $Inv(true) = Q$ of the unique location covers the state space.

Thus, dynamical systems are the class of HA with a single location and no transition.

5.1.2 Symmetric Uncertainty

Recall that in Section 2.3, the set of trajectories $\mathcal{T}$ of HA $\mathcal{A}$ is specified with a flow condition. Formally, for all $\xi \in \mathcal{T}$ and any $t \in [0, \xi.dur]$,

$$\frac{d}{dt}\xi(t).X \in F(\xi(t).X),$$

for some Lipschitz continuous set-valued function $F : val(X) \mapsto 2^{val(X)}$. Let $L > 0$ denote the Lipschitz constant of F. For simplicity, we say that the set-valued function F specifies the set of trajectories $\mathcal{T}$. In many dynamical systems, the set-valued function F has additional topological properties. We introduce a particular class of such functions.

Definition 5 (Symmetric uncertainty)**.** *A dynamical system $\mathcal{A}$ has r*-symmetric uncertainty *for some $r > 0$ if there exists a function $f : val(X) \mapsto \mathbb{R}^{|X|}$ such that for all $\mathbf{x} \in val(X)$,*

$$F(\mathbf{x}) = B_r(f(\mathbf{x})),$$

where F is a set-valued function specifies $\mathcal{A}$.

Roughly, the flow condition of a dynamical system with symmetric uncertainty is specified by a set-valued function F, which maps to a ball around the value of a single-valued function f. The radius of the ball is a constant $r > 0$. It is the same as saying $F(\mathbf{x}) = f(\mathbf{x}) + B_r(0)$ where the sum is in the sense of Minkowski.

5.1.3 Shrink Concretization

We introduce (r, q)-shrink concretization, which transforms a dynamical system with symmetric uncertainty to a dynamical system specified by differential equations. Then, in Lemma 10-12, we show that the reachable set of the original system can be bounded by the reachable set of its (r, q)-shrink concretization.

Definition 6 (Shrink concretization)**.** *For a dynamical system $\mathcal{A} = \langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T} \rangle$, a dynamical system $\mathcal{B} = \langle V', \mathcal{L}', Q', \Theta', Grd', Inv', \mathcal{T}' \rangle$ is a (r, q)-*shrink concretization *of $\mathcal{A}$ for some $r, q \geq 0$ if:*

(i) $V' = V, \mathcal{L}' = L, Q' = Q, Grd' = Grd, Inv' = Inv$,

(ii) for each $\mathbf{x} \in val(X)$, $F'(\mathbf{x}) = \mathsf{Shrink}(F(\mathbf{x}), r)$, where F is specifies $\mathcal{T}$ and F' specifies $\mathcal{T}'$, and

(iii) the initial sets of the two dynamical system satisfy $\mathsf{Shrink}(\Theta, q) \subseteq \Theta' \subseteq \Theta$.

Recall that in Chapter 2 we define $\mathsf{Shrink}(S, \gamma) \triangleq \{x | x \in S \wedge (\min_{y \in \partial S} |y - x|) \geq \gamma\}$ as the set S shrunk by γ. That is, a (r, q)-shrink concretization $\mathcal{B}$ of $\mathcal{A}$ has the same set of variables as $\mathcal{A}$. Compared to $\mathcal{A}$, system $\mathcal{B}$'s behaviors are restricted in two directions. First, the trajectories of $\mathcal{B}$ are specified by a set-valued function (F'), which is obtained by the set-valued function specifies $\mathcal{A}$'s trajectories (F) point-wisely shrunk by r. Second, the initial set of $\mathcal{B}$ is contained by the initial set of $\mathcal{A}$ with a difference no more than q. If $\mathcal{A}$ has symmetric uncertainty, we have the following proposition.

Proposition 8. *For any a dynamical system $\mathcal{A}$ with r-*symmetric uncertainty*, for any $q > 0$ smaller than the radius of its initial set, let $\mathcal{B}$ be the (q, r)-shrink concretization of $\mathcal{A}$, then $\mathcal{B}$ is a dynamical system for which there exists f_B such that all the trajectories ξ of $\mathcal{B}$ satisfy the differential equation $\frac{d}{dt}\xi(t).X = f_B(\xi(t).X)$.*

The proposition follows Definition 5 and Definition 6.

5.1.4 Shift Abstraction

Definition 7 (Shift abstraction)**.** *For a dynamical system* $\mathcal{A} = \langle V, \mathcal{L}, Q, \Theta, Grd, Inv, \mathcal{T}\rangle$*, a dynamical system* $\mathcal{B} = \langle V', \mathcal{L}', Q', \Theta', Grd', Inv', \mathcal{T}'\rangle$ *is a* δ-shift abstraction *of* $\mathcal{A}$ *for some* $\delta, k \geq 0$ *if:*

(i) $V' = V, \mathcal{L}' = L, Q' = Q, Grd' = Grd, Inv' = Inv,$

(ii) the set of trajectories $\mathcal{T}_{\mathcal{A}}$ *and* $\mathcal{T}_{\mathcal{B}}$ *are specified by the same set-valued function* F*, and*

(iii) the initial set of $\mathcal{B}$ *is the reach set of* $\mathcal{A}$ *at time* δ*,* $\Theta' = \mathsf{Reach}_{\mathcal{A}}(\delta)$*.*

That is, $\mathcal{B}$ is a δ-shift abstraction of $\mathcal{A}$ if it has all the same set of variables and flow condition (specified by F) as $\mathcal{A}$. However, the initial set of $\mathcal{B}$ is the reach set of $\mathcal{A}$ at time δ. We denote the δ-shift abstraction of $\mathcal{A}$ by $\mathcal{B} \triangleq \mathcal{S}_\delta(\mathcal{A})$.

Proposition 9. *For a dynamical system* $\mathcal{A}$ *and its shift abstraction* $\mathcal{B} = \mathcal{S}_\delta(\mathcal{A})$*, for any* $t_2 > t_1 \geq \delta$ *,*

$$\begin{aligned} \mathsf{Reach}_{\mathcal{A}}(t) &= \mathsf{Reach}_{\mathcal{B}}(t-\delta), and \\ \mathsf{Reach}_{\mathcal{A}}(t_1, t_2) &= \mathsf{Reach}_{\mathcal{B}}(t_1-\delta, t_2-\delta). \end{aligned}$$

We omit the proof because it is standard. Roughly, the δ-shift abstraction $\mathcal{B}$ is obtained by masking the behavior of $\mathcal{A}$ upto time δ, but nothing else.

So far in Section 5.1, we formally defined dynamical system with r-symmetric uncertainty and (r, q)-shrink concretization. In the next section, we show that the reach set of a dynamical system $\mathcal{A}$ can be upper and lower bounded by the reach set of its (r, q)-shrink concretization.

5.2 Reach set of shrink concretization

In this section we discuss the relation between the reach set of a symmetric uncertain dynamical system $\mathcal{A}$ and that of its shrink concretization $\mathcal{B}$. Our result shows that

$$\mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_2) \subseteq \mathsf{Reach}_{\mathcal{A}}(T) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_1)$$

with some constant $M_1, M_2 > 0$. In addition, by some choice of small enough initial set difference q and time bound $T > 0$, $M_1 - M_2$ can be made arbitrarily small. First, we show the later half of the argument.

Lemma 10. *Let dynamical system $\mathcal{B}$ be the (q, r)-shrink concretization of dynamical system $\mathcal{A}$. Let $L > 0$ be the Lipschitz constant of set-valued function $F_{\mathcal{B}}$ which specifies the set of trajectories of $\mathcal{B}$. For any $T > 0$,*

$$\mathsf{Reach}_{\mathcal{A}}(0, T) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(0, T), M_1), \tag{5.1}$$

where $M_1 \triangleq (q + rT)e^{LT}$.

Proof. From Definition 6, $\mathcal{A}$ and $\mathcal{B}$ share the same variables (V) and state space Q. The distance between the initial sets satisfies $d(\Theta, \Theta') \leq q$. Let $\mathcal{T}_{\mathcal{A}}$ denote the set of trajectories of $\mathcal{A}$ specified by the set-valued function $F_{\mathcal{A}}$. Let $\mathcal{T}_{\mathcal{B}}$ denote the set of trajectories of $\mathcal{B}$ specified by the set-valued function $F_{\mathcal{B}}$.

Fix any $\mathbf{x} \in \mathsf{Reach}_{\mathcal{A}}(0, T)$, there exists $\xi \in \mathcal{T}_{\mathcal{A}}$ and a $t \in [0, T]$ such that $\xi(0).X \in \Theta$ and $\xi(t).X = \mathbf{x}$. From the flow condition of ξ, we know that for all $s \in [0, t]$, $\frac{d}{ds}\xi(s).X \in F_{\mathcal{A}}(\xi(s).X)$.

Recall that $\mathsf{Proj}_{F_{\mathcal{B}}(\xi(s).X)}(\frac{d}{ds}\xi(s).X)$ denotes the set of elements in the set $F_{\mathcal{B}}(\xi(s).X)$ that is closest to the derivative of ξ. We define a function $\tau : [0, t] \mapsto \mathbb{R}^{|X|}$ such that for all $s \in [0, t]$,

$$\tau(s) \in \mathsf{Proj}_{F_{\mathcal{B}}(\xi(s).X)}(\frac{d}{ds}\xi(s).X). \tag{5.2}$$

That is, $\tau(s)$ is an arbitrary element in the set $F_{\mathcal{B}}(\xi(s).X)$ that is closest to the derivative of ξ. From Definition 6, we have $F_{\mathcal{B}}(\xi(s).X) = \mathsf{Shrink}(F_{\mathcal{A}}(\xi(s).X), r)$. That is, for $\frac{d}{dt}\xi(s).X \in F_{\mathcal{A}}(\xi(s).X)$, there always exists an element in $F_{\mathcal{B}}(\xi(s).X)$ within r distance from $\frac{d}{dt}\xi(s).X$. Thus,

$$|\tau(s) - \frac{d}{ds}\xi(s).X| \leq r. \tag{5.3}$$

Define a differentiable function $\xi' : [0, t] \mapsto Q$ such that

(i) $\xi'(0).X \in \Theta'$ such that $|\xi'(0).X - \xi(0).X| \leq q$, and

(ii) for any $s \in [0, t]$,

$$\frac{d}{ds}\xi'(s).X \in \mathsf{Proj}_{F_{\mathcal{B}}(\xi'(s).X)}(\tau(s)). \tag{5.4}$$

From Definition 6, $d(\Theta, \Theta') \leq q$, so there exists $\xi'(0).X$ satisfies (i) above. At any time $s \in [0, t]$, the derivative of ξ' is some element of the set $F_{\mathcal{B}}(\xi'(s).X)$ that is closest to $\tau(s)$. From Equation (5.4), using the property of projection, we have $\frac{d}{ds}\xi'(s).X \in F_{\mathcal{B}}(\xi'(s).X)$. It follows that $\xi' \in \mathcal{T}_{\mathcal{B}}$ and $\xi'(t).X \in \mathsf{Reach}_{\mathcal{B}}$.

From Equation (5.2), $\tau(s) \in F_{\mathcal{B}}(\xi(s).X)$. From Lipschitz condition of $F_{\mathcal{B}}$, we have

$$d(F_{\mathcal{B}}(\xi'(s).X), F_{\mathcal{B}}(\xi(s).X)) \leq L|\xi'(s).X - \xi(s).X|.$$

Combined with Equation (5.4), we have

$$\left|\frac{d}{ds}\xi'(s).X - \tau(s)\right| \leq L|\xi'(s).X - \xi'(s).X|. \tag{5.5}$$

So far, for any $\mathbf{x} \in \mathsf{Reach}_{\mathcal{A}}(0, T)$, we construct a trajectory (ξ') of $\mathcal{B}$. Next, we will prove that $|\xi'(t) - \mathbf{x}| \leq M_1$.

By writing ξ and ξ' in integral form, we have

$$\begin{aligned}
|\xi'(t).X - \xi(t).X| &= \left|\xi'(0).X + \textstyle\int_0^t \frac{d}{ds}\xi'(s).X ds - \xi(0).X - \int_0^t \frac{d}{ds}\xi(s).X ds\right| \\
&\leq |\xi'(0).X - \xi(0).X| + \left|\textstyle\int_0^t \left(\frac{d}{ds}\xi'(s).X - \frac{d}{ds}\xi(s).X\right) ds\right| \\
&\leq q + \textstyle\int_0^t \left|\frac{d}{ds}\xi'(s).X - \frac{d}{ds}\xi(s).X\right| ds \\
&\leq q + \textstyle\int_0^t \left|\frac{d}{ds}\xi'(s).X - \tau(s)\right| ds + \int_0^t \left|\tau(s) - \frac{d}{ds}\xi(s).X\right| ds
\end{aligned}$$

Substitute Equation (5.3) and (5.5) into the above equation, we have

$$|\xi'(t).X - \xi(t).X| \leq q + rt + \int_0^t L|\xi'(s).X - \xi'(s).X| ds. \tag{5.6}$$

By Gronwall-Bellman inequality, we have

$$|\xi'(t).X - \xi(t).X| \leq (q + rt)e^{Lt} \leq (q + rT)e^{LT} = M_2. \tag{5.7}$$

It follows that $\xi(t).X \in B_{(q+r)Te^{LT}}(\xi'(t).X)$. Recall that $\xi(t).X$ is selected arbitrarily in $\mathsf{Reach}_{\mathcal{A}}(0,T)$ and that $\xi'(t).X \in \mathsf{Reach}_{\mathcal{B}}(0,T)$. Thus, $\mathsf{Reach}_{\mathcal{A}}(0,T) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(0,T), (q+rT)e^{LT})$ follows.

□

Corollary 11. *Let dynamical system $\mathcal{B}$ be the (q,r)-shrink concretization of dynamical system $\mathcal{A}$. Let $L > 0$ be the Lipschitz constant of set-valued function $F_{\mathcal{B}}$ which specifies the set of trajectories of $\mathcal{B}$. For any $T > 0$,*

$$\mathsf{Reach}_{\mathcal{A}}(T) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_1). \tag{5.8}$$

In Lemma 10, we show that, given a (r,q)-shrink concretization $\mathcal{B}$ of dynamical system $\mathcal{A}$, there exists a bound M_1 such the reach set of the concretization $\mathcal{B}$ expanded by M_1 guarantees to contain the reach set of $\mathcal{A}$. Corollary 11 follows Lemma 10, where the reach sets at time T are studied instead of reach sets up to time T. Next, we show that there exists a lower bound M_2 such that the reach set of $\mathcal{A}$ at least contain the reach set of $\mathcal{B}$ bloated by M_2.

Lemma 12. *Let dynamical system $\mathcal{B}$ be the (q,r)-shrink concretization of dynamical system $\mathcal{A}$ with r-symmetric uncertainty. Let $L > 0$ be the Lipschitz constant of $f_{\mathcal{B}}$. Define $M_2 \triangleq rT - rLT^2e^{LT}$. For small enough $T > 0$ such that $M_2 > 0$, the following holds:*

$$\mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_2) \subseteq \mathsf{Reach}_{\mathcal{A}}(T), \tag{5.9}$$

Proof. We assume that q, T are selected such that $M_2 > 0$. Fix any $\xi \in \mathcal{T}_{\mathcal{B}}$ with $\xi.dur \geq T$. Let $u : [0,T] \mapsto B_r(0) \subseteq 2^{|X|}$ be a measurable function which we will specify later. Define a function $\xi' : [0,T] \mapsto Q$ as following: (i) $\xi'(0).X = \xi(0).X$, and (ii) $\frac{d}{dt}\xi'(t).X = f_{\mathcal{B}}(\xi'(t).X) + u(t)$. Specified a measurable $u(t)$ for $t \in [0,T]$, $\xi'(t)$ is well defined on $t \in [0,T]$. From the type of $u : [0,T] \mapsto B_r(0)$, it follows that for any $t \in [0,T]$, $|u(t)| \leq r$. Thus, for any t, it follows that

$$\frac{d}{dt}\xi'(t).X = f_{\mathcal{B}}(\xi'(t)) + u(t) \in B_r(f_{\mathcal{B}}(\xi'(t).X)) = F_{\mathcal{A}}(\xi'(t).X).$$

Recall that $\xi'(0).X \in \Theta_{\mathcal{B}} \subseteq \Theta_{\mathcal{A}}$. Thus, for any choice of u of the type defined, we have $\xi'(t) \in \mathcal{T}_{\mathcal{A}}$. Next, we compute the distance between the two trajectory ξ and ξ'. Noticing that ξ and ξ' share the same initial state, we have

$$\begin{aligned} \xi'(T).X - \xi(T).X &= \xi'(0).X - \xi(0).X + \textstyle\int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X) + u(t)\right) dt \\ &= \textstyle\int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X)\right) dt + \int_0^T u(t)dt. \end{aligned} \tag{5.10}$$

From the Lipschitz property of f_B, we have

$$|f_B(\xi'(t).X) - f_B(\xi(t).X)| \leq L|\xi'(t).X - \xi(t).X|. \tag{5.11}$$

It follows that

$$|\xi'(T).X - \xi(T).X| \leq \int_0^T L|\xi'(t).X - \xi(t).X|dt + \int_0^T |u(t)|dt$$

By Gronwall-Bellman's inequality, we get

$$|\xi'(T).X - \xi(T).X| \leq rTe^{LT}.$$

Combine the above iniquity with Equation (5.11), we have

$$\begin{aligned} \left|\textstyle\int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X)\right) dt\right| &\leq \textstyle\int_0^T L|\xi'(T).X - \xi(T).X|dt \\ &\leq rLT^2e^{LT}. \end{aligned} \tag{5.12}$$

Recall Equation (5.10)

$$\xi'(T).X = \xi(T).X + \int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X)\right) dt + \int_0^T u(t)dt.$$

Define a point $\mathbf{x} \triangleq \xi(T).X + \int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X)\right) dt$. Then, $\mathbf{x}$ is a point defined by $\xi(T).X$ added a bounded vector $\int_0^T \left(f_{\mathcal{B}}(\xi'(t).X) - f_{\mathcal{B}}(\xi(t).X)\right) dt$. From Equation (5.12), we know that $\mathbf{x}$ should belongs to the ball $B_{rLT^2e^{LT}}(\xi(T).X)$, as shown in Figure 5.1. Recall that u can be chosen arbitrarily as a measurable function with the type $[0,T] \mapsto B_r(0)$. It follows that the vector $\int_0^T u(t)dt$ can be chosen arbitrarily in the ball $B_{rT}(0)$. From the assumption that $M_2 > 0$, we have $rT > rLT^2e^{LT}$. Thus for any $\mathbf{y} \in B_{rT-rLT^2e^{LT}}(\xi(T).X) = B_{M_2}(\xi(T).X)$, there exists a u such that the trajectory ξ' defined by u reaches $\mathbf{y}$. Thus, we conclude $\xi'(T).X$ can be chosen arbitrarily in the ball $B_{M_2}(\xi(T).X)$. $\square$

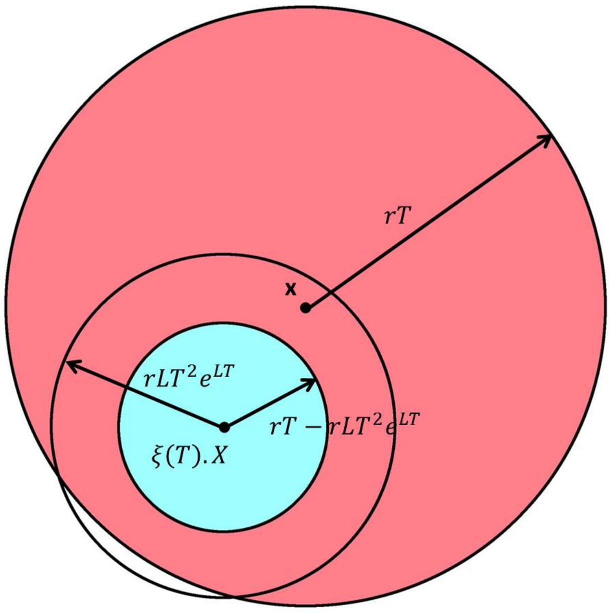


Figure 5.1: An illustration of the reach set of $\xi'(T).X$

The lemma relies on an assumption $M_2 > 0$. Indeed, $rLT^2e^{LT} \sim O(T^2)$ while $rT \sim O(T)$. Thus by choosing small enough $T > 0$, $M_2 = rT - rLT^2e^{LT}$ can be made positive.

So far in this section, we show that:

$$\mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_2) \subseteq \mathsf{Reach}_{\mathcal{A}}(T) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{B}}(T), M_1).$$

The difference between the constant M_1 and M_2 suggests the precision of this bounds. We have,

$$M_1 - M_2 = (q + rT)e^{LT} - rT + rLT^2e^{LT} = qe^{LT} + rT(e^{LT} - 1 + LTe^{LT}).$$

It follows that as $q, T \to 0$, $M_1 - M_2 \to 0$. In the next section, we use this result to compute an overapproximation of the reach set for symmetric uncertain dynamical system. We show that the overapproximation can be made arbitrarily tight.

5.3 Reachability of Symmetric Uncertain Dynamical System

In this section, we present a reachability algorithm that can compute overapproximations the reach set of a dynamical system to arbitrary precision.

5.3.1 Reachability Algorithm

We introduce Algorithm 4 for computing an overapproximation of the reach set of a dynamical system $\mathcal{A}$ with r-symmetric uncertainty. The Algorithm takes five inputs: (i) a dynamical system $\mathcal{A}$, (ii) the measure r of the symmetric uncertainty (of $\mathcal{A}$), (iii) sample period δ, (iv) the time bound T, and (v) precision parameter ϵ. We assume that the time bound T is always divisible by the sample period δ. We define the number of samples to be $k \triangleq \frac{T}{\delta}$.

```
1  $R \leftarrow \emptyset$; $\mathcal{B} \leftarrow \mathcal{A}$; $q \leftarrow 0$;
2  $M_2 \leftarrow r\delta - rL\delta^2 e^{L\delta}$;
3  for $i = 0 : k-1$ do
4      $\mathcal{C} \leftarrow shrink(\mathcal{B}, r)$;
5      $(R_{0T}, R_T) \leftarrow reach(\mathcal{C}, \delta, \epsilon)$;
6      $M_1 \leftarrow (q + r\delta)e^{L\delta}$;
7      $q \leftarrow M_1 - M_2 + \epsilon$;
8      $R \leftarrow R \cup \mathsf{Expand}(R_{0T}, M_1)$;
9      $\Theta \leftarrow \mathsf{Expand}(R_T, M_2 - \epsilon)$;
10     $\mathcal{B} \leftarrow set\Theta(\mathcal{B}, \Theta)$;
11 end
12 return $R$;
```

Algorithm 4: Algorithm to compute $\mathsf{Reach}_{\mathcal{A}}(0, T)$

In this algorithm, the overapproximation of $\mathsf{Reach}_{\mathcal{A}}(0, T)$ is computed iteratively on the intervals $[i\delta, (i+1)\delta]$ for $i = 0, 1, \ldots, k-1$.

Initially (line **??**), R is set to empty set, $\mathcal{B}$ is set to $\mathcal{A}$ and q is set to 0. The constant parameter M_2 is set to $r\delta - rL\delta^2 e^{L\delta}$. In each iteration (line 3-11), $R, \mathcal{B}$ and q get reset (line 7, 8 and 10). At the beginning of each iteration, dynamical system $\mathcal{C}$ is defined by shrink the dynamics of $\mathcal{B}$ (line 4). That is $\mathcal{C}$ is identical to $\mathcal{B}$ except that the differential inclusion specifies $\mathcal{C}$ is derived by the differential inclusion of $\mathcal{B}$ shrunk by r. Because $\mathcal{B}$ has r-symmetric uncertainty as $\mathcal{A}$, $\mathcal{C}$ is a dynamical system specified by differential equations.

Recall that in in Chapter 3, we present an algorithm that computes the reach set of a HA specified by differential equations with arbitrary accuracy (Theorem 1 and 2). In line 5, we use the algorithm presented to compute two sets R_{0T} and R_T, such that

$$\begin{aligned} \mathsf{Reach}_{\mathcal{C}}(0,\delta) \subseteq R_{0T} \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}}(0,\delta),\epsilon), \\ \mathsf{Reach}_{\mathcal{C}}(\delta) \subseteq R_T \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}}(\delta),\epsilon). \end{aligned} \tag{5.13}$$

In line 6, the bloating parameter M_1 are defined. In line 7, the variable q is set to $M_1 - M_2 + \epsilon$. In the next line, R is added the set $R_{0,T}$ expanded by M_1. In line 9-10, we define a new $\mathcal{B}$ by setting its initial set to be Θ , where the set Θ is obtained by bloating the set R_T by q amount.

The algorithm returns a set R and a number q. We will discuss the soundness and completeness of this algorithm in the next section.

5.3.2 Soundness and Completeness

In this section, we show that Algorithm 4 computes overapproximation of $\mathcal{A}$ upto arbitrary precision. That is, for any $d > 0$, there always exists some $\epsilon, \delta > 0$ such that

$$\mathsf{Reach}_{\mathcal{A}}(0,T) \subseteq R \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{A}}(0,T), d),$$

where R is computed by Algorithm 4.

First we prove that in each iteration of a run of Algorithm 4, $\mathcal{C}$ is indeed a (r,q)-shrink concretization of the shift abstraction of $\mathcal{A}$.

Lemma 13. *In a run of Algorithm 4, let $\mathcal{C}^{(j)}$ denote the value of $\mathcal{C}$ assigned in iteration $i = j$. Let $q^{(j)}$ denote the value of q at the beginning of the iteration $i = j$. Then, for each iteration $i = 0, \dots, k-1$, $\mathcal{C}^{(i)}$ is a $(r, q^{(i)})$-shrink concretization of the $i\delta$-shift abstraction of $\mathcal{A}$, $\mathcal{S}_{i\delta}(\mathcal{A})$.*

Proof. Let $\mathcal{B}^{(j)}$ denote the valuation of $\mathcal{B}$ at the beginning of the iteration $i = j$. For any j, $\mathcal{B}^{(j)}$ is specified by the same differential inclusion as $\mathcal{A}$. The differential equation specifies $\mathcal{C}^{(j)}$ is obtained by the differential inclusion of $\mathcal{B}^{(i)}$ shrunk by r (line 4). It follows that differential equation specifies $\mathcal{C}^{(j)}$ is always obtained by the differential inclusion of $\mathcal{A}$ shrunk by r. Note that shift abstraction does not change the dynamics. It follows that $\mathcal{C}^{(i)}$ and $\mathcal{S}_{i\delta}(\mathcal{A})$ satisfies condition (ii) of the definition of (r,q)-shrink concretization (Definition 6). Thus, $\mathcal{C}^{(i)}$ is a $(r,q^{(i)})$-shrink concretization of $\mathcal{S}_{i\delta}(\mathcal{A})$ if and only if the initial set of $\mathcal{C}^{(i)}$, denoted $\Theta^{(i)}$, is contained by the initial set of $\mathcal{S}_{i\delta}(\mathcal{A})$ with difference no more than $q^{(i)}$ (condition (iii) in Definition 6). The initial set of $\mathcal{S}_{i\delta}(\mathcal{A})$ is defined by $\mathsf{Reach}_{\mathcal{A}}(i\delta)$. Thus, for proving the lemma, it is sufficient to prove the following hold for all i:

$$\Theta^{(i)} \subseteq \mathsf{Reach}_{\mathcal{A}}(i\delta) \subseteq \mathsf{Expand}(\Theta^{(i)}, q^{(n)}). \tag{5.14}$$

We prove this argument by induction.
Base case: for $i=0$, we have $q^{(0)}=0$ and $\mathcal{B}^{(0)}=\mathcal{A}$. $\Theta^{(0)}$ is the same as the initial set of $\mathcal{A}$. It follows that $\Theta^{(0)}=\mathsf{Reach}_{\mathcal{A}}(0)$. Equation (5.14) holds for $i=0$.
Induction: If for $i=j$, $\mathcal{C}^{(j)}$ is a $(r,q^{(j)})$-shrink concretization of $\mathcal{S}_{j\delta}(\mathcal{A})$, we will prove Equation (5.14) holds for $i=j+1$. Let $M_1^{(j)}$ be the value of M_1 defined in iteration $i=j$. It follows Corollary 11 and Lemma 12 that

$$\mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}(\delta), M_2) \subseteq \mathsf{Reach}_{\mathcal{S}_{j\delta}(\mathcal{A})}(\delta) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}(\delta), M_1^{(j)}). \tag{5.15}$$

From Definition 7, $\mathsf{Reach}_{\mathcal{S}_{j\delta}(\mathcal{A})}(\delta) = \mathsf{Reach}_{\mathcal{A}}((k+1)\delta)$. From line 5, the set $R_T^{(j)}$ satisfies:

$$\mathsf{Reach}_{\mathcal{C}^{(j)}}(\delta) \subseteq R_T^{(j)} \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}, \epsilon) \tag{5.16}$$

Expand the first half of the above equation by the amount of $M_1^{(j)}$, we get

$$\mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}(\delta), M_1^{(j)}) \subseteq \mathsf{Expand}(R_T^{(j)}, M_1^{(j)}). \tag{5.17}$$

Expand the later half of of Equation (5.16) by the amount of $M_2-\epsilon$, we have

$$\mathsf{Expand}(R_T^{(j)}, M_2-\epsilon) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}, M_2). \tag{5.18}$$

Combining Equation (5.15), (5.17) and (5.18), we get

$$\mathsf{Expand}(R_T^{(j)}, M_2-\epsilon) \subseteq \mathsf{Reach}_{\mathcal{A}}((j+1)\delta) \subseteq \mathsf{Expand}(R_T^{(j)}, M_1^{(j)}). \tag{5.19}$$

From line 9 of Algorithm 4, we have $\Theta^{(j+1)} = \mathsf{Expand}(R_T^{(j)}, M_2 - \epsilon)$. From line 7 we get $q^{(j+1)} = M_1^{(j)} - M_2 + \epsilon$. Substitute $\Theta^{(j+1)}$ and $q^{(j+1)}$ into Equation (5.19), we finally derive:

$$\Theta^{(j+1)} \subseteq \mathsf{Reach}_{\mathcal{S}_{j\delta}(\mathcal{A})}(\delta) \subseteq \mathsf{Expand}(\Theta^{(j+1)}, q^{(j+1)}).$$

It follows that, Equation (5.14) holds for $i = j + 1$.

By induction, we show that Equation (5.14) holds for any $i = 0, \ldots, k-1$. Thus the lemma follows. □

Combined the above lemma with Lemma 10 and 12, we can prove that Algorithm 4 is both sound and complete.

Theorem 3. *For a dynamical system $\mathcal{A}$ with r-symmetric uncertainty, for any time bound $T > 0$, for any sample period $\delta > 0$, and any precision parameter $\epsilon > 0$,*

$$\mathsf{Reach}_{\mathcal{A}}(0, T) \subseteq R,$$

where R is computed by Algorithm 4 using $\mathcal{A}, r, T, \delta, \epsilon$.

Proof. By the Proposition 9, for $T = k\delta$, the reach set of $\mathcal{A}$ can be decomposed as the union of its shift abstractions'.

$$\mathsf{Reach}_{\mathcal{A}}(0, T) = \cup_{i=0}^{k-1} \mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta).$$

We denote $R_{0T}^{(j)}$ and $M_1^{(j)}$ as the value of R_{0T} and M_1 assigned in the iteration $i = j$. We will prove that for each iteration $i = j$,

$$\mathsf{Expand}(R_{0T}^{(j)}, M_1^{(j)}) \supseteq \mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta).$$

In Lemma 13, it is proved that $\mathcal{C}^{(j)}$ is the $(r, q^{(j)})$-shrink concretization of $\mathcal{S}_{i\delta}(\mathcal{A})$. It follows Lemma 10 that $\mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}(0, \delta), M_1^{(j)}) \supseteq \mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta)$. From line 5, it follows that $R_{0T}^{(j)} \supseteq \mathsf{Reach}_{\mathcal{C}^{(j)}}(0, \delta)$. Thus we get $\mathsf{Expand}(R_{0T}^{(j)}, M_1^{(j)}) \supseteq \mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta)$. From line 8, we have that $R \supseteq \mathsf{Expand}(R_{0T}^{(j)}, M_1^{(j)})$ for any $j = 0, \ldots, k-1$. It follows that $R \supseteq \mathsf{Reach}_{\mathcal{A}}(0, T)$. □

The above theorem establishes the soundness of Algorithm 4. Next, we show that the overapproximation can be made arbitrarily tight. In the following lemma, we show that the value of q can be made arbitrarily small.

Lemma 14. *Fix any dynamical system $\mathcal{A}$ with r-symmetric uncertainty and any time bound $T > 0$. For any precision requirement $d > 0$, there exist small enough $\delta > 0$ and $\epsilon > 0$, such that for all iteration $i = 0, 1, \ldots, k-1$,*

$$q^{(i)} < d,$$

where $q^{(j)}$ denotes the assignment of q in the iteration $i = j$, in a run of Algorithm 4 with parameters $\mathcal{A}, r, T, \delta, \epsilon$.

Proof. We denote $M_1^{(j)}$ and $q^{(j)}$ as the values of variables M_1 and q assigned in the iteration $i = j$. Note that $q^{(i)}$ is increasing in i. It is sufficient to show that $q^{(k-1)} < d$. Before any iteration, q is initialized to 0. The parameter $M_2 = r\delta(1 - L\delta e^{L\delta})$ is constant during a run of the algorithm. In the iteration $i = 0$, from line 6 we have $M_1^{(0)} = r\delta e^{L\delta}$. From line 7, $q^{(0)} = r\delta(e^{L\delta} - 1 + L\delta e^{L\delta}) + \epsilon$. In the iteration $i = j+1$ iteration, $M_1^{(j+1)}$ is computed by

$$M_1^{(j+1)} = (q^{(j)} + r\delta)e^{L\delta} \tag{5.20}$$

It follows that $q^{(j+1)}$ satisfies a recurrence relation:

$$q^{(j+1)} = M_1^{(j+1)} - M_2 + \epsilon = e^{L\delta}q^{(j)} + r\delta(e^{L\delta} - 1 + LTe^{L\delta}) + \epsilon.$$

Solving the recurrence relation with initial condition $q^{(0)} = r\delta(e^{L\delta} - 1 + L\delta e^{L\delta}) + \epsilon$, we have:

$$q^{(k-1)} = \frac{e^{k\delta L} - 1}{e^{L\delta} - 1}\left(r\delta(e^{L\delta} - 1 + L\delta e^{L\delta}) + \epsilon\right).$$

Note that $e^{L\delta} \sim L\delta$ and that $k\delta = T$. It follows that

$$q^{(k-1)} \sim \frac{e^{LT} - 1}{L\delta}\left(r\delta(L\delta + L\delta e^{L\delta}) + \epsilon\right) \sim \delta r(e^{LT} - 1)(e^{L\delta} + 1) + \epsilon\frac{e^{LT} - 1}{L\delta}.$$

It follows that by selecting $\epsilon \sim O(\delta^2)$, $q^{(k-1)} \sim O(\delta)$. Thus, the values of q can be made arbitrarily small ($q < d$) by choosing small enough $\delta, \epsilon > 0$. □

With Lemma 14, we are ready to state our last theorem which establishes the completeness of Algorithm 4.

Theorem 4. *For a dynamical system $\mathcal{A}$ with r-symmetric uncertainty, for any time bound $T > 0$, for any precision requirement $d > 0$, there exists* *small enough sample period $\delta > 0$ and precision parameter $\epsilon > 0$, such that*

$$R \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{A}}(0, T), d),$$

where R is computed by Algorithm 4 using $\mathcal{A}, r, T, \delta, \epsilon$.

Proof. Fix any $d > 0$. By Proposition 9, for $T = k\delta$, the reach set of $\mathcal{A}$ can be decomposed as the union of its shift abstractions'.

$$\mathsf{Reach}_{\mathcal{A}}(0, T) = \cup_{i=0}^{k-1} \mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta).$$

We denote $R_{0T}^{(j)}$ and $M_1^{(j)}$ as the value of R_{0T} and M_1 assigned in the iteration $i = j$. We will prove that for each iteration $i = j$,

$$\mathsf{Expand}(R_{0T}^{(j)}, M_1^{(j)}) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta), d).$$

In Lemma 13, it is proved that $\mathcal{C}^{(j)}$ is the $(r, q^{(j)})$-shrink concretization of $\mathcal{S}_{i\delta}(\mathcal{A})$. It follows that

$$\mathsf{Reach}_{\mathcal{C}^{(j)}}(0, \delta) \subseteq \mathsf{Reach}_{\mathcal{S}_{j\delta}(\mathcal{A})}(0, \delta). \tag{5.21}$$

From line 5, we have $R_{0T}^{(j)} \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{C}^{(j)}}(0, \delta), \epsilon)$. It follows that

$$\mathsf{Expand}(R_{0T}^{(j)}, M_1^{(j)}) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{S}_{j\delta}(\mathcal{A})}(0, \delta), \epsilon + M_1^{(j)}). \tag{5.22}$$

Notice that $M_1^{(j)} = (q^{(j-1)} + r\delta)e^{L\delta}$. Note that $q^{(j)}$ is increasing in j. In Lemma 14, we show that the final value of $q^{(k-1)}$ can be made arbitrarily small by choosing sufficiently small δ, ϵ. Thus, we can chose sufficiently small δ, ϵ such that $(q^{(k-1)} + r\delta)e^{L\delta} + \epsilon < d$. It follows that, for each $i = 0, \ldots, k-1$,

$$\mathsf{Expand}(R_{0T}^{(i)}, M_1^{(i)}) \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta), d)$$

Let R be the set computed by Algorithm 4. It follows that

$$\begin{aligned} R &= \cup_{i=0}^{k-1} \mathsf{Expand}(R_{0T}^{(i)}, M_1^{(i)}) \subseteq \cup_{i=0}^{k-1} \mathsf{Expand}(\mathsf{Reach}_{\mathcal{S}_{i\delta}(\mathcal{A})}(0, \delta), d) \\ &\subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{A}}(0, T), d). \end{aligned}$$

The theorem thus follows. □

5.4 Summary

In this chapter we define the notions of shrink concretization and shift abstraction. The reach set of $\mathcal{A}$ is related to the reach set of its shrink concretization as well as its shift abstraction. In Section 5.3, we presented a reachability algorithm for a dynamical system $\mathcal{A}$ with symmetric uncertainty. The Algorithm runs in iteration. In each iteration, a (r, q)-shrink concretization $\mathcal{C}$ of the shift abstraction of $\mathcal{A}$ is defined. Specifically, $\mathcal{C}$ is specified by differential equation, which facilitates reachability computation. Then, the overapproximation of reach set of $\mathcal{A}$ is obtained by manipulating the reach set of $\mathcal{C}$.

This reachability algorithm can handle a general class of dynamical systems specified with non-linear differential inclusions. We prove that, for any $d > 0$, by choosing small enough sample period δ and precision ϵ, the set R returned by the algorithm satisfies $\mathsf{Reach}_{\mathcal{A}}(0, T) \subseteq R \subseteq \mathsf{Expand}(\mathsf{Reach}_{\mathcal{A}}(0, T), d)$.

CHAPTER 6

CONCLUSIONS

In this thesis, we presented simulation-based reachability algorithms for different classes of hybrid systems. In Chapter 3, an algorithm compute an overapproximation of the reach set of a given hybrid automaton $\mathcal{A}$. The trajectories of hybrid automaton $\mathcal{A}$ is specified by a set of differential equations. We also show that the overapproximation our algorithm computes can be made arbitrarily tight. Then, in Chapter 4 we introduce the implementation of the tool **HTV** and present experiment results on hybrid system benchmarks including room heating systems, navigation systems, satellites systems and engine control systems. Finally, in Chapter 5 we present an algorithm computes the overapproximation of the reach set of a dynamical system $\mathcal{A}$, where the trajectories of $\mathcal{A}$ are specified by a set of differential inclusions.

6.1 Contribution

For a hybrid system $\mathcal{A}$, trajectories of which are specified by non-linear differential equations, we show that there exists a cover of the initial set Θ of $\mathcal{A}$ (we call (δ, γ, k-representative cover)) such that executions start from the same cover start from initial states close to each other and visit the same sequence of locations in bounded time. We assume the (δ, γ, k)-representative covers are computable. Then, given a simulation trace β from a state in a cover, we can compute an overapproximation of the time-bounded reachable states of any execution of $\mathcal{A}$ whose initial states in the same cover as β. This enables us to verify bounded time safety property. The proposed algorithm combines the dynamic information from β and the static information from the model $\mathcal{A}$.

We have implemented a MATLAB tool called *Hybrid Trace Verifier (***HTV***)* that implments this methodology and uses traces generated from Mathwork's

Simulink/Stateflow (SLSF). For static analysis **HTV** translates the SLSF models to hybrid automata using HyLink.

For a dynamical system $\mathcal{A}$ specified by non-linear differential inclusions, we presented an algorithm that can compute the bounded reach set of $\mathcal{A}$ with arbitrary precision. The algorithm constructs a sequence of shrink concretizations $\mathcal{C}^{(i)}$ of $\mathcal{A}$. The precise reach sets of $\mathcal{C}^{(i)}$ can be computed using algorithm presented in Chapter 3. Then, precise reach sets of $\mathcal{A}$ can be obtained using the reach sets of $\mathcal{C}^{(i)}$.

The reachability algorithms presented in this thesis can handle non-linear dynamics, thus can potentially apply to a variety of practical systems. For both algorithm presented, the soundness and completeness are formally proved.

6.2 Future Directions

The results from this thesis suggest several possible directions. A natural extension of both algorithms is to use other classes of trajectory annotations besides Lipschitz dynamics. Possible candidates of the annotation include contraction metrics and incremental stability and Lyapunov as studied in [37]. For the algorithm for hybrid system with deterministic transitions, an extension can be to consider non-deterministic transitions. For this objective, a simulation tree instead of a single simulation trace from an initial state might be useful for capturing behaviors of the true executions. I will also be interesting to study dynamical system specified by differential inclusion with non-symmetric uncertainties.

REFERENCES

[1] C. Lee, “Vw issues software upgrade to resolve gearbox problems, shies away from recall,” 2012. [Online]. Available: http://autonews.gasgoo.com/china-news/vw-issues-software-upgrade-to-resolve-gearbox-prob-120313.shtml

[2] K. Poulsen, “Software bug contributed to blackout,” 2004. [Online]. Available: http://www.securityfocus.com/news/8016

[3] S. Mitra, T. Wongpiromsarn, and R. M. Murray, “Verifying cyber-physical interactions in safety-critical systems,” *IEEE Security & Privacy*, vol. 99, no. PrePrints, p. 1, 2013.

[4] A. Gupta, R. Majumdar, and A. Rybalchenko, “From tests to proofs,” in *TACAS*, 2009, pp. 262–276.

[5] N. E. Beckman, A. V. Nori, S. K. Rajamani, and R. J. Simmons, “Proofs from tests,” in *Proceedings of the 2008 international symposium on Software testing and analysis*, ser. ISSTA ’08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1390630.1390634 pp. 3–14.

[6] N. Lynch, R. Segala, and F. Vaandrager, “Hybrid I/O automata,” in *Hybrid Systems III*, ser. Lecture Notes in Computer Science, vol. 1066. Springer-Verlag, 1996, pp. 496–510.

[7] S. Mitra, “A verification framework for hybrid systems,” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2007.

[8] T. MathWorks, “Simulink 7,” March 2008.

[9] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata*, ser. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005, also available as Technical Report MIT-LCS-TR-917.

[10] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo, “A step towards verification and synthesis from simulink/stateflow models,” in *HSCC*, 2011.

[11] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[12] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[13] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *ACM Symposium on Theory of Computing*, 1995, pp. 373–382.

[14] R. Bagnara, P. M. Hill, and E. Zaffanella, "The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Science of Computer Programming*, vol. 72, no. 1–2, pp. 3–21, 2008.

[15] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Proceedings of the 8th International Conference onHybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, M. Morari and L. Thiele, Eds. Springer Berlin / Heidelberg, 2005, vol. 3414, pp. 291–305.

[16] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal techniques for reachability analysis of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 52, pp. 26–38, 2007.

[17] P. Prabhakar, V. Vladimerou, M. Viswanathan, and G. E. Dullerud, "Verifying tolerant systems using polynomial approximations," in *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, ser. Real Time Systems Symposium. IEEE Computer Society, 2009, pp. 181–190.

[18] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 1, pp. 110–122, 1997.

[19] A. Chutinan and B. Krogh, "Computational techniques for hybrid system verification," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 64–75, Jan. 2003.

[20] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past hytech," in *Proceedings of the 8th International Conference on Hybrid Systems: Computation and Control*, 2005, pp. 258–273.

[21] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification*, ser. LNCS. Springer, 2011.

[22] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *TCS*, vol. 410, pp. 4262–4291, September 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1609208.1609591

[23] A. Kanade, R. Alur, F. Ivancic, S. Ramesh, S. Sankaranarayanan, and K. Shashidhar, "Generating and analyzing symbolic traces of simulink/stateflow models," in *CAV*, 2009.

[24] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *TACAS*, 2011.

[25] O. Bouissou and M. Martel, "Grklib: a guaranteed runge kutta library," in *IMACS*, 2006.

[26] A. Puri and P. Varaiya, "Decidability of hybrid systems with rectangular differential inclusions," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, D. Dill, Ed. Springer Berlin Heidelberg, 1994, vol. 818, pp. 95–104. [Online]. Available: http://dx.doi.org/10.1007/3-540-58179-0_46

[27] T. A. Henzinger and R. Majumdar, "Symbolic model checking for rectangular hybrid systems," in *Proceedings of the Sixth International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000).* LNCS 1785, Springer-Verlag, 2000, pp. 142–156.

[28] K.-D. Kim, S. Mitra, and P. R. Kumar, "Bounded epsilon-reachability of linear hybrid automata with a deterministic and transversal discrete transition condition," in *CDC*, 2010.

[29] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Hybrid Systems: Computation and Control.* Springer, 2000, pp. 73–88.

[30] G. Pace and G. Schneider, "Model checking polygonal differential inclusions using invariance kernels," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 2937, pp. 110–121. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24622-0_11

[31] E. Michael, "Continuous selections i," *Ann. of Math*, vol. 63, no. 2, pp. 361–382, 1956.

[32] G. Lafferriere, G. J. Pappas, and S. Sastry, "O-minimal hybrid systems," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 13, pp. 1–21, 2000.

[33] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. Dullerud, "Stormed hybrid systems," in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 136–147.

[34] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal: a tool suite for automatic verification of real-time systems," in *Hybrid Systems III*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1996, vol. 1066, pp. 232–243.

[35] A. S. Morse, "Supervisory control of families of linear set-point controllers - part 1: Exact matching," *IEEE Trans. Automat. Contr*, vol. 41, pp. 1413–1431, 1998.

[36] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 1992.

[37] S. P.S.Duggirala and M.Viswanathan, "Verification of annotated models from executions," in *International Conference on Embedded Software*, 2013.

[38] A. Fehnker and F. Ivancic, "Benchmarks for hybrid systems verification," in *In Hybrid Systems: Computation and Control (HSCC 2004) (2004*. Springer, 2004, pp. 326–341.

[39] T. Johnson, J. Green, S. Mitra, R. Dudley, and R. S. Erwin, "Verifying satellite rendezvous and conjunction avoidance: A formal approach to autonomy in space," in *International Conference on Formal Methods (FM)*, 2012.