Sampling-based Falsification and Verification of Controllers for Continuous Dynamic Systems

Peng Cheng and Vijay Kumar

GRASP Lab, University of Pennsylvania, {chpeng, kumar}@grasp.upenn.edu

Abstract: In this paper, we present a sampling-based verification algorithm for continuous dynamic systems with uncertainty due to adversaries, unmodeled disturbance inputs, unknown parameters, or initial conditions. The algorithm attempts to find inputs (and resulting trajectories) that falsify the specifications of the system thus providing examples of bad inputs to the system. The system is said to be verified if the algorithm cannot find falsifying inputs.

The main contribution of the paper is the analysis of the effects of discretization of the state and input spaces that are inherent to sampling-based techniques. We derive conditions that guarantee resolution completeness. These provide sufficient, although conservative, conditions for verifying Lipschitz continuous (but possibly non smooth) dynamic systems without known analytical solutions. We analyze the effects of transformations of the input and state space on these conditions. The main results of this paper are illustrated with several simple examples.

1 Introduction

Software-enabled control of dynamical systems finds applications not only in robotics, but also in manufacturing, fly-by-wire systems, air-traffic control, medical instrumentation, and biotechnology. There is currently no systematic approach to verifying controllers for systems with continuous input and state spaces except for a very special class of simple systems for which analytical solutions are readily available. Indeed, if we exclude this special class of systems, the verification problem is generally undecidable [1].

The *falsification* problem is similar to the *motion planning* problem. In the former, one tries to find the disturbance or adversarial inputs that result in trajectories which violate system specifications, for example, safety. In the later, we find inputs that guide the system to a state that satisfies specifications for the goal set. In our approach, the *verification* problem is solved by showing the absence of falsifying inputs or trajectories. Thus, a system is said to be verified if there are no falsifying inputs. Analogously, in motion planning, one can try to prove no motion plans exist to reach the goal set.

Because general verification problems are undecidable, semi-decidable approximation algorithms have been designed. Most of these algorithms [2, 8, 18] over-approximate the reachable set to check the safety. However such algorithms are limited in their ability to handle complex dynamics in high dimensions. Recently, motivated by the successful application of sampling-based techniques in motion planning [6, 11, 17, 15, 14, 16] and the strong similarity between motion planning and falsification, researchers have developed algorithms [4, 9, 12] that use sampled controls to under-approximate the continuous search space to quickly find counter examples to show that the system is not safe. However, there is no principled way to verify system properties.

The paper presents a sampling-based verification algorithm for Lipschitz continuous but possibly non smooth systems. The verification is achieved by using sampling-based falsification algorithms, which iteratively construct solutions with sampled controls to falsify the given safety specification. Similar approaches have been proposed for linear systems [10] and for hybrid systems [5]. Because sampling-based control algorithms discretize the input and state spaces and approximate the set of trajectories (and therefore the reachable space), it is necessary to establish a relationship between the discretization of these spaces and the approximation of the reachable set, and quantify the confidence level associated with the falsification or verification result. The main goal of this paper is a set of conditions that establishes this connection. The basic result is that a proper choice of sampling dispersion (in input and state spaces) and an appropriate sampling algorithm will ensure that every falsifying control with a finite time horizon will be approximated within a desired level of fidelity by sampled controls in finite time.

This work is closely connected to previous work in which conditions for *resolution completeness* of sampling-based motion planning with differential constraints were established for the first time [7]. It is showed [7] that solutions to motion planning problems for dynamic systems will always be approximated by sample controls in finite time. Of course, no guarantees are offered for problems for which no solutions exist. The key idea is to use Lipschitz conditions on motion equations to develop resolution-complete algorithms. The proof for resolution-completeness relies on establishing that the reachable state set is densely covered by the states reached by sample controls.

In the same spirit, we introduce a relaxed problem, in which the safety specification is relaxed with a given tolerance to enlarge the set of falsifying controls. A resolution-complete (RC) falsification algorithm is designed to approximate falsifying controls for the relaxed problem. If no solutions are found for the relaxed problem, then there exist no falsifying controls for the original problem and the system is verified. This is illustrated schematically in Fig. 1. The shaded region represents the unsafe set for the original problem. The unsafe set of the relaxed problem shown as the set inside the dashed line includes all points which are in the ϵ neighborhood of the unsafe set of the original problem. All falsifying controls for the original problem turn into falsifying controls with violation ϵ for the relaxed problem. If all trajectories

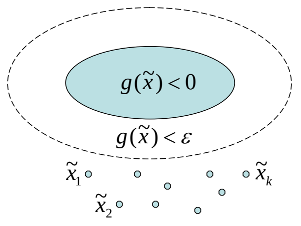


Fig. 1. Verification by falsification. $\tilde{x}$ denotes the system trajectory and the function $g(\tilde{x})$ defines the specification set or the unsafe set. Tolerance ϵ defines the relaxed problem. A trajectory is said to be falsifying for the relaxed problem if $g(\tilde{x}) < \epsilon$.

$\{\tilde{x}_i\}$ constructed from the RC falsification algorithm are outside of unsafe set of the relaxed problem, then the system is said to be verified.

The organization of this paper is as follows. First, we formally define the dynamic system and the falsification and verification problems in Section 2. Section 3 provides a framework for sampling-based falsification and discusses the complexity of the algorithms. In Section 4, we present the verification algorithm through RC falsification and analyze the effects of scaling and transformation on RC conditions. Several examples are used to illustrate the application of the proposed algorithm in Section 5.

2 Falsification and Verification Problems

In this section, we formally define the dynamic systems of interest, the basic assumptions, and the falsification and verification problems. We use standard notation found in most books on systems theory (see, e.g., [13]).

The dynamic system is described as follows:

$$\dot{x} = \frac{dx}{dt} = f(x, u), x \in X, u \in U, \tag{1}$$

in which $X \subset \Re^n$ is the *state space* and $U \subset \Re^m$ is the *input space*. We assume that x and u are nondimensionalized. X and U are given the structure of a metric space using the infinity norm. We will assume that these sets are bounded and there exist D_u and D_x such that $\|u - u'\| < D_u$ for any $u, u' \in U$ and $\|x - x'\| < D_x$ for any $x, x' \in X$. We assume that the motion equation satisfies the Lipschitz condition with respect to state and input. There exist positive constants L_x and L_u such that:

$$\|f(x, u) - f(x', u')\| \leq L_x \|x - x'\| + L_u \|u - u'\| \tag{2}$$

for any $x, x' \in X$ and $u, u' \in U$. There also exists real constant $D_f > 0$ such that $\|f(x, u)\| < D_f$ for any $x \in X$ and $u \in U$.

The control space $\mathcal{U}$ (a function space) is assumed to include all piecewise constant controls $\tilde{u} : [0, t_f] \rightarrow U$. We will also assume that each input is only applied over a constant interval, δt, and there is a positive integer, k, such that $t_f = k\delta t$. Both these assumptions are for simplicity. The extension to more general function spaces is described in [7].

Given a control $\tilde{u} : [0, t_f] \to U$ and a state $x_0 \in X$, the trajectory of the control from x_0 is

$$\tilde{x}(\tilde{u}, x_0, t) = x_0 + \int_0^t f(\tilde{x}(\tau), \tilde{u}(\tau))d\tau. \tag{3}$$

$\tilde{x}(\tilde{u}, x_0)$ is also used to denote the trajectory from x_0 as a function of time. The set X_{init} includes all possible initial states of the system. The trajectory space $\tilde{\mathcal{X}}$ for the problem is a function space defined by:

$$\tilde{\mathcal{X}} = \{\tilde{x}(\tilde{u}, x) \mid \tilde{u} \in \mathcal{U}, x \in X_{\text{init}}\}, \tag{4}$$

which could be generalized to the trajectory space for the system by replacing X_{init} with X. Assume that $\tilde{x} : [0, t_1] \to X$ and $\tilde{x}' : [0, t_2] \to X$ are two trajectories in $\tilde{\mathcal{X}}$ and $t_2 \geq t_1$, the metric for the trajectory space is

$$\rho_x(\tilde{x}, \tilde{x}') = |t_1 - t_2| + \max\Big(\max_{t\in[0,t_1]} \|\tilde{x}(t) - \tilde{x}'(t)\|, \max_{t\in[t_1,t_2]} \|\tilde{x}'(t) - \tilde{x}(t_1)\|\Big). \tag{5}$$

This metric can be easily shown to satisfy the standard metric axioms.

The unsafe set or the *specification set* is characterized by a continuous function $g : \tilde{\mathcal{X}} \to R$. If there exists $\tilde{x}(\tilde{u}, x) \in \tilde{\mathcal{X}}$ such that $g(\tilde{x}) < 0$, then the system is unsafe. Note that both spatial and temporal constraints can be incorporated in such functions. The function $g(\tilde{x})$ is assumed to be Lipschitz continuous with respect to $\tilde{x}$. For any $\tilde{x}, \tilde{x}' \in \tilde{\mathcal{X}}$

$$|g(\tilde{x}) - g(\tilde{x}')| \leq L_b \rho_x(\tilde{x}, \tilde{x}'). \tag{6}$$

Finally, we will only consider problems with finite time horizons. Further we require this time horizon, D_T, to be a integer multiple of δt. In other words, $D_T = K\delta t$ for some positive integer K. We are now in a position to define the verification and falsification problems.

Definition 1. Falsification problem: *Find a falsifying control $\tilde{u} \in \mathcal{U}$ and a state $x_0 \in X_{\text{init}}$ such that $g(\tilde{x}(\tilde{u}, x_0)) < 0$.*

Definition 2. Verification problem: *Verify that there does not exist any falsifying controls $\tilde{u} \in \mathcal{U}$ with a state $x_0 \in X_{\text{init}}$ such that $g(\tilde{x}(\tilde{u}, x_0)) < 0$.*

Definition 3. Falsifying control with *violation ϵ***:** *A falsifying control $\tilde{u} \in \mathcal{U}$ and a state $x_0 \in X_{\text{init}}$ such that $g(\tilde{x}(\tilde{u}, x_0)) < -\epsilon$ for some $\epsilon > 0$.*

To facilitate the proof in Section 4, we will define relaxed version of the falsification problem below.

Definition 4. ϵ-relaxed falsification problem *Find a falsifying control $\tilde{u} \in \mathcal{U}$ and state $x_0 \in X_{\text{init}}$ such that its trajectory $\tilde{x}(\tilde{u}, x_0)$ satisfies $g(\tilde{x}(\tilde{u}, x_0)) < \epsilon$.*

3 Sampling-based falsification algorithm

Because our verification algorithm is achieved through falsification, we will first describe a sampling-based falsification algorithm, which will be converted into a verification algorithm by RC conditions in Section 4.1. There are many sampling-based motion planning algorithms that can be used to design falsification algorithms. However, because our goal is to use the falsification algorithm for verification, we will only use the most basic algorithm and focus instead on its use for verification and not describe the different variants and heuristics of sampling-based algorithms.

3.1 The basic falsification algorithm

To solve the falsification problem described in Section 2, we will assume that we are given a state sampling dispersion bound α_x, and an input sampling *dispersion* bound α_u. Dispersion is the radius of the largest empty ball in a given sample point set [19]. A finite sample state set $\mathcal{S}_x \subset X_{\text{init}}$ is chosen with dispersion less than the given α_x and a finite sample input set $\mathcal{S}_u \subset U$ is determined with dispersion less than α_u. These bounds are illustrated in Figure 2, in which dashed lines show the largest empty balls for the infinity norms, and small dots in (a) and (b) represent sample states in $\mathcal{S}_x$ and sample inputs in $\mathcal{S}_u$ respectively. The algorithm iteratively constructs a search graph

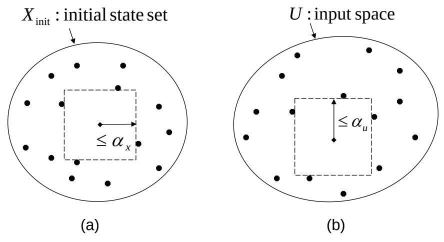


Fig. 2. The given dispersion bounds α_x and α_u are used to determine $\mathcal{S}_x$ and $\mathcal{S}_u$.

using sample inputs in $\mathcal{S}_u$ from sample states in $\mathcal{S}_x$. The search graph is a directed graph. Every node n corresponds to a state $x(n) \in X$. If a sample input $u_l \in \mathcal{S}_u$ is applied for a duration δt from a node, n_k, to generate a control $\tilde{u} \in \mathcal{U}$, resulting in a trajectory segment $\tilde{x}(\tilde{u}, x(n_k))$, then this input u_l is said to have been *applied* for the node n_k. If all inputs in $\mathcal{S}_u$ have been applied for a node, then that node is called *expanded*.

For a problem with the unsafe set described by $g(\tilde{x}) < 0$, the sampling-based falsification algorithm is as follows.

1. Initialize the algorithm: Initialize the search graph by associating each state in $\mathcal{S}_x$ with a new node. There are no edges in the graph.

2. Select an unexpanded node in the search graph: If every node in the search graph is expanded, then the algorithm returns.
3. Generate a trajectory segment with an unapplied sampled input: Choose an unapplied input from $\mathcal{S}_u$ for the selected node. Apply the sample input on the selected node to generate a trajectory segment. Evaluate the function $g(\tilde{x})$ with respect to the current trajectory.
4. Update the search graph: If $g(\tilde{x}) \geq 0$ and the search depth is no larger than K (described in Section 2), then the final state is associated with a new node in the search graph and a new edge is inserted from the selected node to the new node; otherwise, a falsifying control is returned.
5. Iterate from Step 2 until no node is selected.

3.2 The falsification algorithm with state space discretization

In many algorithms, such as [3], state space discretization is used to decrease the computational complexity of the algorithm by restricting the maximal number of nodes in the search graph.

The discretization is governed by the dispersion bound α_x. The state space X is discretized into a finite number of non overlapping sets so that the maximal distance between any two states in a set is less than α_x. Every set allows at most one node in the search graph. If it contains one node, it is called *occupied*; otherwise, it is called *empty*. Thus before inserting a node for a new state at the end of the trajectory segment of duration δt, a check is performed to see if the discrete set in which the new state is in, is occupied or not. If it is occupied by an existing node, then no new nodes are added. However, a new edge must still be inserted from the selected node to the existing node.

State space discretization directly affects the computations that need to be performed for falsification. Because one input is applied on one unexpanded node in each iteration, the upper bound on the number of computations will be the product of the maximal number of nodes in the search graph and the number of sample inputs in $\mathcal{S}_u$. The size of $\mathcal{S}_u$ is $|\mathcal{S}_u| = O\left([D_u/\alpha_u]^m\right)$.

If we do not discretize the state space, every sample input from a node in the search graph can potentially generate a new node. Therefore, the number of nodes in a search graph starting from a node in K steps is bounded by summing a geometric series: $O\left((|\mathcal{S}_u|^{K+1} - 1)/(|\mathcal{S}_u| - 1)\right)$. Potentially we can have $|\mathcal{S}_x| = O([\frac{D_x}{\alpha_x}]^n)$ disjointed search graphs. Thus the number of iterations of the basic algorithm without discretizing the state space is:

$$O\left([D_x/\alpha_x]^n\,[D_u/\alpha_u]^{m(K+1)}\right). \tag{7}$$

If we do discretize the state space, the number of nodes in the search graph is bounded by the number of non overlapping sets in the partition. The number of sets is $O([\frac{D_x}{\alpha_x}]^n)$ and the number of iterations of the algorithm is

$$O\left(\,[D_u/\alpha_u]^m\,[D_x/\alpha_x]^n\right). \tag{8}$$

Thus state space discretization greatly reduces the upper bound on the number of iterations.

4 A Resolution Complete Algorithm for Verification

In this section, the falsification algorithm in Section 3 is first converted into a verification algorithm by adapting RC conditions for motion planning with differential constraints in Section 4.1. The choice of dispersion bounds with respect to the computation budget, the effects of state and input space transformation on algorithm parameters are respectively provided in Sections 4.2 and 4.3.

We will first define ϵ-resolution completeness.

Definition 5. ϵ-Resolution Complete (ϵ-RC) falsification algorithm *Given a falsification problem, if there exists a falsifying control $\tilde{u}$ with violation $\epsilon > 0$, then an ϵ-RC falsification algorithm will find a falsifying control $\tilde{u}'$ in finite time.*

In other words, if there exists $\tilde{u}$ and $x_0 \in X_{\text{init}}$ such that $g(\tilde{x}(\tilde{u}, x_0)) < -\epsilon$, then an ϵ-RC falsification algorithm will find a control $\tilde{u}'$ and $x_0' \in X_{\text{init}}$ such that $g(\tilde{x}(\tilde{u}', x_0')) < 0$.

4.1 Verification through RC falsification

To solve the verification problem, we simply run the algorithms in Sections 3.1 and 3.2 on the ϵ-relaxed falsification problem. It will be shown in the following that if α_x and α_u are appropriately chosen, then all falsifying controls for the original falsification problem will be approximated and returned as solutions of the relaxed problem. If no solution is returned, then the system is verified.

Note: The function describing the unsafe set for the ϵ-relaxed falsification problem is $g'(\tilde{x}) = g(\tilde{x}) - \epsilon < 0$. Therefore, if $g'(\tilde{x}) = g(\tilde{x}) - \epsilon < 0$ in Step 4 of the algorithm in Section 3.1, a falsifying control will be returned.

Theorem 1. *For a given $\epsilon > 0$, if an ϵ-RC algorithm does not find a solution with respect to the ϵ-relaxed falsification problem in finite time, then the system in the original problem is verified.*

Proof. Every falsifying control for the original problem is a falsifying control with violation ϵ for the ϵ-relaxed problem, which will be approximated and returned as a solution to the relaxed problem in finite time by an ϵ-RC algorithm. Conversely, if no solution is returned for ϵ-relaxed problem, then the system is verified. $\diamond$

Recall α_x and α_u are the dispersion bounds for sampling in X and U. The following theorem provides the choice of algorithm parameters to ensure that the falsification algorithm in Section 3 is ϵ-resolution complete.

Theorem 2. *If the dispersion bounds satisfy the RC inequality* $\lambda\alpha_x + \gamma\alpha_u < \sigma$ *with*

$$\sigma = \frac{\epsilon}{L_b}, \lambda = \frac{e^{L_x \delta t(K+1)} - 1}{e^{L_x \delta t} - 1}, \gamma = L_u \delta t e^{L_x \delta t} \frac{e^{L_x \delta t K} - 1}{e^{L_x \delta t} - 1}, \tag{9}$$

then the falsification algorithms in Section 3 are ϵ*-RC falsification algorithms.*

Proof. The proof will show that under the conditions in the above theorem, every falsifying control $\tilde{u}$ with violation ϵ will be approximated and returned by the algorithm. The proof follows a similar reasoning as in [7]. Instead of presenting the proof, we present the main intuition behind the idea.

As shown in Fig. 3, sampling in the control space means only an approximate solution $\tilde{u}'$ of a falsifying control $\tilde{u}$ can be returned from our sampling-based algorithm (see (a)). Furthermore, the state space discretization and state sampling in X_{init} result in discontinuities in the trajectories $\hat{x}(\tilde{u}', x_0')$ in our search graph. There is a discontinuity in (c) because x_{new} and $x(n_e)$ are not the same point and the initial state x_0 is approximated by x_0' in (b). The main observation is that the dispersion bounds α_x and α_u bound the variation of initial states, the trajectory discontinuities, and the control mismatches. Because the system is Lipschitz continuous and the time horizon is finite, for any $\tilde{u}$ and $x_0 \in X_{\text{init}}$ there always exist (adapted from Theorem 2.5 in [13]) $\tilde{u}'$ and $x_0' \in X_{\text{init}}$ such that

$$\rho_x(\tilde{x}(\tilde{u}, x_0), \hat{x}(\tilde{u}', x_0')) < \lambda\alpha_x + \gamma\alpha_u, \tag{10}$$

in which λ and γ are given as above. Recall from (6) the function g has a

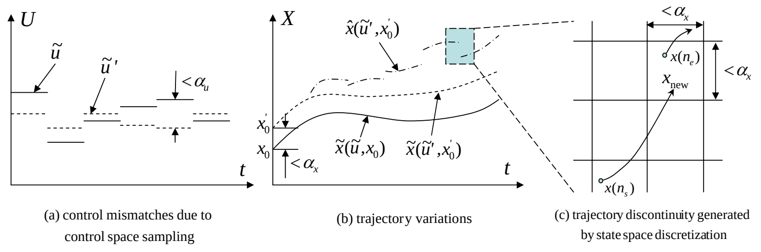

U
$\tilde{u}$
$\tilde{u}'$
$<\alpha_u$
t
(a) control mismatches due to control space sampling
X
$\hat{x}(\tilde{u}',x_0')$
x_0'
x_0
$<\alpha_x$
$\tilde{x}(\tilde{u},x_0)$
$\tilde{x}(\tilde{u}',x_0')$
t
(b) trajectory variations
$<\alpha_x$
$x(n_e)$
x_{new}
$<\alpha_x$
$x(n_s)$
(c) trajectory discontinuity generated by state space discretization

Fig. 3. The intuition of RC conditions

Lipschitz constant L_b. Therefore, if there exists a falsifying trajectory $\tilde{x}(\tilde{u}, x_0)$ with violation ϵ, an approximation $\tilde{x}(\tilde{u}', x_0')$ that satisfies

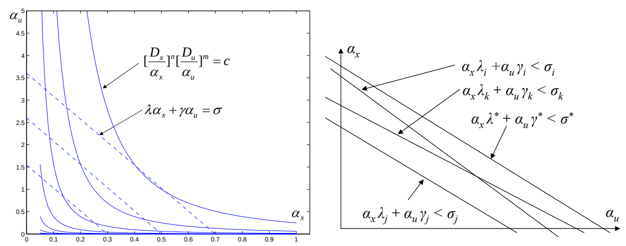
$$\rho_x(\tilde{x}, \tilde{x}') < \epsilon / L_b, \tag{11}$$

will be a falsifying control. The conditions in the theorem immediately follow by requiring the right side of (10) be less than the right side of (11). ⋄

4.2 Choice of dispersion bounds

The computational burden is determined by first determining an upper bound T_{iter} on running time for each iteration and the upper bound on the number of iterations. Since the later directly depends on the dispersion bounds α_x and α_u (see (8)), the RC inequality in Theorem 2 indirectly determines the computations required for the ϵ-relaxed falsification problem.

This is illustrated in Fig. 4 (a) for a simple example, in which $D_u = D_x = 1$, $m = 1$, $n = 2$, $\lambda = 0.278$, and $\gamma = 1.43$. For a given T_{iter}, the solid lines are iso-cost curves representing a fixed computational cost for different choices of dispersion bounds. The closer the iso-cost lines are to the origin, the higher the required computational cost. The straight dashed lines represents the RC inequality in Theorem 2 for different choices of relaxation ϵ. The closer the lines are to the origin, the smaller the relaxation ϵ. For a given computational



(a) (b)

Fig. 4. (a) Selection of algorithm parameters with respect to computational resources (b) Comparison of different RC inequalities

budget we can, in principle, find the minimally relaxed falsification problem, for which the RC inequality line will be tangent to the iso-cost curve with the given computational budget allowing us to determine the dispersion bounds.

4.3 Transformations on X and U

We have assumed that the underlying spaces are metric spaces. Often it is necessary to introduce scaling transformations to non dimensionalize the system so that we are not affected by using non homogeneous coordinates (for example, Cartesian coordinates and angles) and inputs (for example, torques and forces). But one can also imagine transforming the underlying spaces to take advantage of dimensions along which the dynamic system may evolve slowly (slow time scale) and focus instead on dimensions along which changes happen more rapidly (fast time scale). In what follows, we will explore the effects of transformations allowing for general transformation of X and U.

Consider the following transformation: $x = T_x\bar{x}$, $u = T_u\bar{u}$, and $t = T_c\bar{t}$, in which T_x and T_u are full rank square matrices, and T_c is a positive real number. The state space X and input space U are respectively transformed into $\bar{X}$ and $\bar{U}$. If $T_x = \beta I$, $T_u = \beta I$, and $T_c = \beta$ for some real constant $\beta > 0$, then the transformation is called a *uniform scaling* transformation.

The transformed motion equation is

$$\dot{\bar{x}} = \frac{d\bar{x}}{d\bar{t}} = \bar{f}(\bar{x}, \bar{u}) = T_c T_x^{-1} f(T_x \bar{x}, T_u \bar{u}). \tag{12}$$

The Lipschitz constants with respect to the state and input for (1) are

$$L_x = \sup_{x,u} \left\| \frac{\partial f}{\partial x}(x,u) \right\|, \; L_u = \sup_{x,u} \left\| \frac{\partial f}{\partial u}(x,u) \right\|, \tag{13}$$

and for (12) are:

$$L_{\bar{x}} = T_c \sup_{x,u} \left\| T_x^{-1} \frac{\partial f}{\partial x}(x,u) T_x \right\|, L_{\bar{u}} = T_c \sup_{x,u} \left\| T_x^{-1} \frac{\partial f}{\partial u}(x,u) T_u \right\|. \tag{14}$$

Because matrix multiplication does not commute, L_x is different from $L_{\bar{x}}$ for general transformations.

RC inequalities under transformation

Theorem 3. *The RC inequality after the transformation has*

$$\sigma = \frac{\epsilon}{L_b}, \; \lambda = \max(T_c, \|T_x\|_\infty)\|T_x^{-1}\|_\infty \frac{e^{L_{\bar{x}}\bar{\delta t}(\bar{K}+1)} - 1}{e^{L_{\bar{x}}\bar{\delta t}} - 1}, \tag{15}$$

and

$$\gamma = \max(T_c, \|T_x\|_\infty)\|T_u^{-1}\|_\infty L_{\bar{u}} \bar{\delta t} e^{L_{\bar{x}}\bar{\delta t}} \frac{e^{L_{\bar{x}}\bar{\delta t}\bar{K}} - 1}{e^{L_{\bar{x}}\bar{\delta t}} - 1}. \tag{16}$$

Proof. With the given transformation, we have

$$\rho_x(\tilde{x}, \tilde{x}') \leq \max(T_c, \|T_x\|_\infty)\rho_x(\tilde{\bar{x}}, \tilde{\bar{x}}'). \tag{17}$$

The given algorithm parameters $\epsilon_{\bar{x}}$ and $\epsilon_{\bar{u}}$ are described with respect to the new spaces. With these algorithm parameters, for any falsifying control $\tilde{\bar{u}}$ and initial state $\bar{x}_0$, there exists an approximation $\tilde{\bar{u}}$ and state $\bar{x}_0'$ such that

$$\rho_x(\tilde{\bar{x}}(\tilde{\bar{u}}, \bar{x}_0), \hat{\bar{x}}(\tilde{\bar{u}}', \bar{x}_0')) < \epsilon_{\bar{x}} \tfrac{e^{L_{\bar{x}}\bar{\delta t}(\bar{K}+1)} - 1}{e^{L_{\bar{x}}\bar{\delta t}} - 1} + \epsilon_{\bar{u}} L_{\bar{u}} \bar{\delta t} e^{L_{\bar{x}}\bar{\delta t}} \tfrac{e^{L_{\bar{x}}\bar{\delta t}\bar{K}} - 1}{e^{L_{\bar{x}}\bar{\delta t}} - 1} \tag{18}$$

With infinity norms on the state and input space, it can be verified that

$$\epsilon_{\bar{x}} \leq \|T_x^{-1}\|_\infty \epsilon_x, \epsilon_{\bar{u}} \leq \|T_u^{-1}\|_\infty \epsilon_u. \tag{19}$$

Substituting the above inequalities into (18) and requiring the right side of (17) be less than ϵ/L_b will complete the proof. $\diamond$

Corollary 1. *The RC inequality is invariant for any uniform scaling.*

Proof. With any uniform scaling, it can be verified that $\max(T_c, \|T_x\|_\infty) = 1/\|T_x^{-1}\|_\infty = 1/\|T_u^{-1}\|_\infty$, $K = \bar{K}$, $\delta t = \bar{\delta t}$, $L_x = L_{\bar{x}}/\beta$, and $L_u = L_{\bar{u}}/\beta$. Therefore, the same RC inequality coefficients in Theorem 2 will always be derived by substituting these equalities into the RC inequality coefficients in Theorem 3. $\diamond$

Comparison of RC conditions with different transformations

From the above description, we can see that the derived RC inequality might not be invariant for non-uniform scaling, such as transformation $T_x = \beta I$, $T_u = \beta I$, and $T_c = \xi > \beta$. Assume that λ_i, γ_i, and σ_i are coefficients of the derived RC inequality, which are obtained from Transformation i. Let

$$E_i = \{(\alpha_x, \alpha_u) \mid \alpha_x \lambda_i + \alpha_u \gamma_i < \sigma_i, \alpha_x > 0, \alpha_u > 0\}. \tag{20}$$

The inequalities and set E_i are shown in Fig. 4 (b). For Transformations i, j, and k, Transformation i is said to be superior to Transformation j if $E_j \subset E_i$. If $E_i \not\subset E_k$ and $E_k \not\subset E_i$, then Transformation i is neither better nor worse than Transformation k. A transformation can be said to be "optimal" from the standpoint of resolution completeness if the set defined by $\alpha_x \lambda^* + \alpha_u \gamma^* < \sigma^*$ is not the subset of E_i for all other transformations. Again this "optimal" transformation will generate dispersion bounds that are larger so that the maximal number of nodes, the size of $\mathcal{S}_u$, and therefore the computational cost will be smaller.

5 Examples

In this section we illustrate the sampling-based falsification and verification methodology, the use of ϵ-relaxation, and transformation of state and input spaces. We choose several simple verification problems that allow easy interpretation. The first problem has parametric uncertainty in inputs while the second problem has parametric uncertainty in the initial state. The third problem incorporates uncertainty in the form of disturbance input functions. The final problem presents an analysis of control policies for pursuit evasion.

5.1 Verification problems

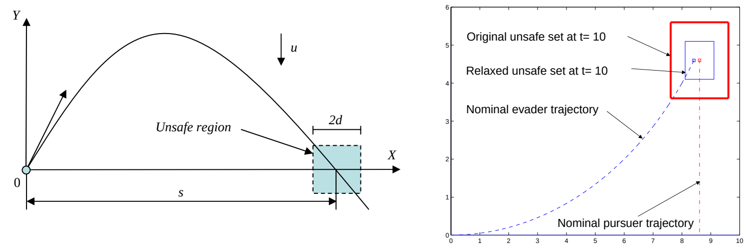
Problem 1: Verification of a system with an uncertain parameter Consider a point mass which moves freely on a plane with constant but unknown external force, u, along the y-axis (see Fig. 5 (a)). The state x of the system includes (p_x, v_x, p_y, v_y) in $X = [0, 15] \times [1, 3] \times [-1, 1] \times [-1, 2]$, which denote the position and velocity along x and y axes respectively. Its motion equation is $\dot{p}_x = v_x$, $\dot{v}_x = 0$, $\dot{p}_y = v_y$, and $\dot{v}_y = u$, in which $u \in U = [5, 15]$ is the system parameter determining the magnitude of the constant input. The system has initial state $x_0 = (0.0, 2.0, 0.0, 1.0)$. The system is safe if the trajectory of the point mass from initial state x_0 always stays outside of an unsafe region (shown shaded in Fig. 5 (a)), which is a square of width $d = 0.5$ with its center at point $(10, 0)$. The function defining the unsafe set[1] is

[1] Recall that we are considering X as a metric space with the infinity norm.

$$g(\tilde{x}(\tilde{u}, x_0)) = \min_t(\|\tilde{x}(\tilde{u}, x_0, t) - [10, 0]^T\|) - 0.5 < 0.$$

It can be verified that $g(\tilde{x})$ satisfies (6) with Lipschitz constant $L_b = 1$.

The verification problem is to check whether the system is safe for all inputs. There is a natural choice for the finite time horizon, D_T. For $t > 0.7$, p_y can be shown to be less than -0.5 and decreasing. Therefore, we choose $D_T = 0.7$. Because analytical solutions are available for this simple system, it is straightforward to show that the system is safe. We will verify this using a sampling-based algorithm in the next subsection.



Y
u
Unsafe region
2d
X
0
s
Original unsafe set at t= 10
Relaxed unsafe set at t= 10
Nominal evader trajectory
Nominal pursuer trajectory

(a) (b)

Fig. 5. Simple verification problems

Problem 2: Verification of a system with an uncertain initial state Consider the autonomous system with no control: $\dot{y} = 0.2y \sin t^2$ and $\dot{t} = 1$. We define the extended state $x = [y, t]^T \in X = [0, 2] \times [0, 10]$. The initial state is unknown, but restricted to lie in the set $X_{\text{init}} = [0, 0.1] \times 0$. The system is considered to be safe if at $t = 9$ seconds, $\|y(t) - 1.0\| > 0.5$. Again the Lipschitz constant L_b for the function g is 1. The time horizon is $D_T = 9$ seconds. We consider ϵ-relaxed problems with $\epsilon = 0.5$ first and then 0.05.

Problem 3: Verification of a system under input disturbances Consider the kinematic model of a UAV whose nominal inputs are constant but are subject to bounded disturbances. The dynamics is characterized as $\dot{x} = (v_0 + v)\cos\theta$, $\dot{y} = (v_0 + v)\sin\theta$, and $\dot{\theta} = w_0 + w$, in which $v_0 = 1$ and $w_0 = 0.1$ are the nominal inputs for the system, $x \in [0, 10]$, $y \in [0, 5]$, and $\theta \in [0, 2\pi]$ are position and orientation, $v \in [-0.01, 0.01]$ and $w \in [-0.001, 0.001]$ denote the disturbances. The system starts from the initial state $(0, 0, 0)$ at time 0. See Fig. 6 (a). The question is whether the system will stay in the 1.0-neighborhood of the goal position $[x_g, y_g]^T = [8.41, 4.60]^T$ under the input disturbance at time 10 seconds. Thus, the system is said to be unsafe if [2] $\|[x, y]^T - [x_g, y_g]^T\| > 1.0$ at $t = 10$. Again, the Lipschitz constant $L_b = 1$. The disturbance control space consists of piecewise-constant controls with $\delta t = 2$ seconds. We will consider ϵ relaxation with $\epsilon = 0.5$.

[2] The infinity norm is used here.

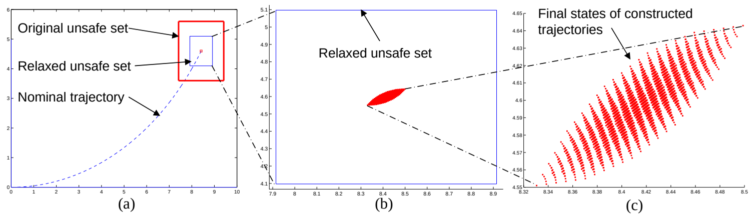


Fig. 6. Verification of the system under input disturbances

Problem 4: Verification of a control policy for the pursuer Consider the UAV in Problem 3 as an evader and a point mass model for a pursuer with position p_x and p_y. The pursuer captures the evader if

$$\|[x(t), y(t)]^T - [p_x(t), p_y(t)]^T\| < 1.0$$

for some t in a finite time horizon of $D_T = 10$ seconds. The UAV has a given nominal control input but with bounded disturbance in the input. An open-loop trajectory ($p_x(t) = 8.61$ and $p_y(t) = 0.46t$) is computed for the pursuer according to the nominal trajectory to achieve capture (see Fig. 5 (b)). The pursuer trajectory is verified if the pursuer can capture the evader over any disturbance from its nominal control. Again, the Lipschitz constant $L_b = 1$. The disturbance control space consists of piecewise-constant controls with $\delta t = 2$ seconds. We will consider ϵ relaxation with $\epsilon = 0.5$.

5.2 RC inequalities under scaling and transformation

The transformation is achieved with following diagonal matrices

$$T_x = \mathrm{Diag}(a_{11}, a_{22}, \cdots, a_{nn}), T_u = \mathrm{Diag}(b_{11}, b_{22}, \cdots, b_{mm}). \tag{21}$$

Problem 1: Because the control is constant, the control space is $\mathcal{U} = \{\tilde{u} \mid \tilde{u}(t) = c, c \in U\}$. Because the algorithm without state space discretization is used and the initial state is a point, RC inequality will be in form $\gamma\alpha_u < \sigma$. We use the ϵ-relaxed falsification problem with $\epsilon = 0.2$. With this ϵ, the ϵ-RC inequalities after four different transformations are listed in Table 1. It can be seen that the RC inequalities are the same for the uniform scaling transformation between 1 and 2 (see Table 1).

Problem 2: Because input space sampling does not exist, RC inequality for this problem will be in form $\lambda\alpha_x < \sigma$. RC inequalities are calculated in Table 2 (a) for a fixed $T_c = 1$.

Problem 3: The ϵ-RC inequalities are calculated in Table 2 (b) with $T_c = 1$ and T_u equal to an identity matrix. The algorithm with state space discretization is used for falsification and verification.

Problem 4: The same ϵ-RC inequalities are obtained as for Problem 3.

Table 1. RC inequalities under different transformations

No.	a_{11}	a_{22}	a_{33}	a_{44}	b_{11}	T_c	$L_{\bar{x}}$	$L_{\bar{u}}$	γ_i	σ_i
1	1	1	1	1	1	1	1	1	1.41	0.2
2	10	10	10	10	10	10	10	10	1.41	0.2
3	10	1	10	1	1	1	0.1	1	7.51	0.2
4	10	100	10	100	1	100	1000	1	767.64	0.2

Table 2. RC inequalities under different transformations

No.	a_{11}	a_{22}	$L_{\bar{x}}$	λ_i	σ_i
1	1.0	1.0	8.2	1.12e32	0.5
2	10.0	1.0	1.0	8.10e4	0.5
3	100.0	1.0	0.28	1.34e3	0.5
4	1000.0	1.0	0.208	7.50e3	0.5

(a)

a_{11}	a_{22}	a_{33}	$L_{\bar{x}}$	$L_{\bar{u}}$	λ_i	γ_i	σ_i
1	1	1	1.01	1	2.81e4	5.61e4	0.5
10	10	1	1.01e-1	1	105.4	190.88	0.5
100	100	1	1.01e-2	1	631.5	1.06e3	0.5
1000	1000	1	1.01e-3	1	6.03e3	1.01e4	0.5

(b)

5.3 Simulation results

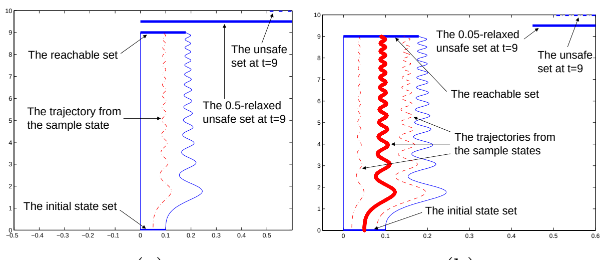
Problem 1: Under Transformation 1 in Table 1, we choose $\alpha_u = 0.141$. A sample input set $\mathcal{S}_u$ with this dispersion bound is $\{5, 5.28, 5.56, \cdots, 15\}$. The system was verified because no solution was returned.

Problem 2: From Table 2 (a), we can see that Transformation 3 yields the best RC inequality in terms of the lowest λ (highest dispersion). The dispersion bound α_x is chosen to be 3.7×10^{-4} to satisfy this inequality. Sample states from X_{init} are $\{0, 7.0 \times 10^{-4}, 1.4 \times 10^{-3}, \cdots, 0.0994, 0.1\}$ are used for simulation. As shown in Fig. 7 (a), the final state of the trajectory from $y = 0.05$ and $t = 0$ is returned by the ϵ-RC falsification problem with $\epsilon = 0.5$, and therefore, the system is not verified.

In order to investigate this problem further, the relaxation tolerance ϵ is reduced to 0.05. For the same transformation, the state sampling dispersion bound α_x is calculated to be 2.5×10^{-5}. Now all the final states of the approximated trajectories are outside of the 0.05-relaxed unsafe set. Therefore, the system is verified. Three sample trajectories are illustrated in Fig. 7 (b).

Problem 3: From Table 2(b), we can see that Transformation 2 has the best RC inequality. We chose dispersion bounds $\alpha_x = 1.1 \times 10^{-3}$ and $\alpha_u = 2.01 \times 10^{-3}$ which satisfy this inequality. The chosen sample input (v, w) with the specified input dispersion is in $\{-0.01, -0.006, -0.002, 0.002, 0.006, 0.01\} \times \{0.0\}$. As shown in Fig. 6 (b) and (c), since the final states of all constructed trajectories do not enter the unsafe region, the system is verified.

Problem 4: The verification algorithm runs with the same choice of the sample input set as in Problem 3. The pursuer trajectory is verified because no disturbance input for the evader is a falsifying control for the relaxed problem. Note that the complexity of the proposed verification for this problem depends



(a) (b)

Fig. 7. Trajectories computed by the verification algorithm for ϵ-relaxed problems

only on the state and input space of the evader. Increasing the number of pursuers does not change the computational cost.

6 Conclusion

In this paper, we proposed a sampling-based verification algorithm based on resolution complete falsification, which involves the iterative construction of solutions that falsify the given safety specification with sampled controls. We derive sufficient conditions for the discretization of the state and input spaces to guarantee that we can find approximations to any falsifying control inputs, if they exist. Thus the paper provides a novel and systematic approach to verifying controllers for continuous dynamic systems.

While the paper presents sufficient conditions for resolution completeness, these conditions are conservative and require a high resolution sampling in state and input spaces for most practical problems. This is because the verification problem is extremely hard. (Recall that the path planning problem (without dynamics) is NP-hard.) We provide a partial solution to this problem by pursuing transformations of input and state spaces that might allow a lower resolution while guaranteeing resolution completeness. This continues to be an area of ongoing research.

Of course heuristics can improve performance by several orders. As shown in the RC inequality in Theorem 2, the complexity of the verification algorithm increases exponentially with the time horizon, and dimension of the state space and input spaces. Thus it is important to prune the search space based on domain knowledge. Our preliminary work in this direction is discussed in [9].

Acknowledgements

We gratefully acknowledge support from NSF grant CNS-0410514 and ONR grant FA8650-04-C-7133.

References

1. R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proccedings of the IEEE*, 88(2):971–984, July 2000.
2. E. Asarin, O. Bournez, T. Dang, and O. Malzer. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems : Computation and Control*. Springer Verlag, 2000.
3. J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.
4. A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *Hybrid Systems : Computation and Control*, Philadelphia, USA, 3 2004.
5. M. Branicky, M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control, and verification of hybrid systems. *IEEE Proc. Control Theory and Applications*. (Accepted).
6. B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *IEEE Int. Conf. Robot. & Autom.*, 2005.
7. P. Cheng. *Sampling-based Motion Planning with Differential Constraints*. PhD thesis, University of Illinois, Urbana, IL, 2005.
8. A. Chutinan and B. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46:1401–1410, 2001.
9. J. M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Proc. Workshop on Algorithmic Foundation of Robotics*, 2004.
10. A. Girard and G. Pappas. Verification using simulation. In *Hybrid Systems : Computation and Control*. Springer Verlag, 2006.
11. D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, 4:495–512, 1999.
12. J. Kapinski, B. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *Hybrid Systems : Computation and Control*. Springer Verlag, 2003.
13. H. Khalil. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 1996.
14. A. Ladd and L. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics and Automation*, 20(2):229–242, April 2004.
15. S. LaValle, M. Branicky, and S. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 24, 2004.
16. S. LaValle and J. K. Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.
17. L.Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.
18. I. Mitchell and C. Tomlin. Overappoximating reachable sets by hamilton-jacobi projections. *J. of Sci. Comput.*, 19, Dec. 2003.
19. H. Niederreiter. *Random Number Generation and Quasi-Monte-Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1992.