Counter-example Guided Learning of Bounds on Environment Behavior

Yuxiao Chen, Sumanth Dathathri, Tung Phan-Minh, and Richard M. Murray
California Institute of Technology
{chenyx, sdathath, tung}@caltech.edu, murray@cds.caltech.edu

Abstract: There is a growing interest in building autonomous systems that interact with complex environments. The difficulty associated with obtaining an accurate model for such environments poses a challenge to the task of assessing and guaranteeing the system's performance. We present a data-driven solution that allows for a system to be evaluated for specification conformance without an accurate model of the environment. Our approach involves learning a conservative reactive bound of the environment's behavior using data and specification of the system's desired behavior. First, the approach begins by learning a conservative reactive bound on the environment's actions that captures its possible behaviors with high probability. This bound is then used to assist verification, and if the verification fails under this bound, the algorithm returns counter-examples to show how failure occurs and then uses these to refine the bound. We demonstrate the applicability of the approach through two case-studies: i) verifying controllers for a toy multi-robot system, and ii) verifying an instance of human-robot interaction during a lane-change maneuver given real-world human driving data.

1 Introduction

In control and decision-making tasks, typically, the system can be divided into the controlled agent and the uncontrolled environment, which is the source of exogenous disturbances and uncertainties. For systems that are safety-critical, given a control policy, it is desirable that we are able to provide guarantees with regard to task fulfillment and safe behavior. In this regard, formal verification allows us to provide strong guarantees about the absence of unsafe behavior for the controlled agent under all possible behaviors of the environment. However, to leverage the power of formal verification it is necessary to obtain a reliable model of the environment. For systems that exhibit complex behavior, accurately modeling the complex environment can be limited by the expressiveness of the model being used, and the amount of data available. In addition, the environment behavior is usually nondeterministic, which may be hard to express.

To overcome this need for explicitly modeling the complete environment, we instead propose an alternative approach that computes *reactive bounds* on the set of feasible behaviors of the environment by leveraging i) a specification for the behavior of the system, and ii) the controller being verified, in addition to the real data collected from naturalistic environment behaviors. The bounds computed here are *reactive* in that they capture the reactiveness of the environment towards the controlled agent, i.e., the set of possible environment behaviors changes with the scenario described by the system states. For the purpose of verification, it often may suffice to compute a conservative bound on the possible behaviors of the unmodeled environment as opposed to learning a complete model of the environment. For example, consider a scenario where two cars are driving perpendicularly towards an intersection with the same distance d to the intersection, and the controlled car (whose controller we seek to verify) drives with a constant velocity v. To verify that there will be no collision (the desired specification), it suffices to bound the velocity of the other car by u such that: $\frac{d+w}{u} < \frac{d}{v}$, where w is the width of the intersection. This guarantees that the uncontrolled car will not enter the intersection before the controlled car leaves the intersection. Here, a conservative bound on the behavior of the uncontrolled agent (environment) suffices to verify the behavior of the controlled agent, without modeling the exact behavior of the environment. However, having bounds on the behavior of the environment that are too loose might result in the controller not being

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

provably safe with regard to the bound, despite being safe with regard to the true environment. In this direction, we present a novel approach where the specification and the controller being verified guide the learning of a reactive bound from data.

Recently, there has been an increased interest in data-driven verification for cyber-physical systems [1, 2, 3]. In [4], a data-driven automated approach is proposed to identify non-converging behaviors in black box control systems. In [5], the authors propose an approach based on Bayesian inference and reachability analysis for verifying the behavior of systems. However, the approach does not decompose the system into the uncontrolled environment and the controller, and is limited to the model class of linear time-invariant systems. In contrast, our approach leverages known policies for the controlled agent to enable verification of their behavior with complex environments. A closely related direction of work is on mining specifications [6, 7, 8] from data. The mined specifications are often used as task specifications, as opposed to being used to verify a given controller. For the case of human-robot interaction, treating it as a multi-agent task and leveraging the influence of the autonomous agent on the (uncontrolled) human has been considered before [9, 10]. In [11], the authors propose an approach that learns a reward function to model the behavior of the uncontrolled agent and then plans for the autonomous agent, leveraging this reward function to model the influence of the autonomous agent on the uncontrolled agent. Here the authors incorporate the environments behavior into the planning phase, while in contrast, we leverage the controlled agent's policy and the desired safety specification for the system to characterize the environment's behavior. [12] demonstrates the benefits of learning the intent of the uncontrolled agent prior to the physical event, and leveraging this for seamless collaboration. In our work, we aim at learning a set of possible environment behaviors that changes with the state, which enables worst-case analysis and is subsequently used for verification of the system. To summarize, our main contributions are:

- A framework for characterizing the behavior of the environment for which the given controller can be certified safe with high probability: given data characterizing the behavior of the environment, the safety-specification and the controller for the system.
- When the controller is inherently unsafe, a feedback mechanism for the controller.
- Experiments demonstrating the efficiency of the proposed approach on a toy multi-agent task, and verifying a controller for an autonomous vehicle performing a lane change while interacting with a human-driven vehicle characterized by real-world data.

The paper is structured as follows. First, we provide a brief introduction to Signal Temporal Logic (STL) and Random Convex Programs (RCP). Then, we provide an overview of the proposed approach, followed by a theoretical analysis. Lastly, we describe our results from two empirical case studies on problems from diverse domains and then conclude.

2 Preliminaries

Consider a dynamical system Σ described by differential or difference equations:

$$x^+ = f(x, u, d), \tag{1}$$

where $x \in \mathcal{X}$ is the state, $u \in \mathcal{U}$ is the control input, $d \in \mathcal{D}$ is environment input, and $\mathcal{X}$, $\mathcal{U}$, $\mathcal{D}$ incorporates the physical limits of the variables. A run of Σ is an indexed family σ consisting of 3-tuples of the form $\sigma_t = (x(t), u(t), d(t))$, satisfying the dynamics equation. If σ is a run of Σ, we will also write $\sigma \models \Sigma$. A run can be infinite or finite with horizon T.

2.1 Signal Temporal Logic

To express desired properties for the system, we use the formalism of *Signal Temporal Logic* (STL) [13], an extension of Linear Temporal Logic to vector-valued signals. For any $a, b \in \mathbb{R}$, we will denote by $[a, b]$ the closed interval $\{x \in \mathbb{R} \mid a \leq x \leq b\}$. STL formulae can be built recursively as:

$$\varphi \triangleq \texttt{True} \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \mathbf{U}_{[a,b]} \psi,$$

where p is an atomic predicate of the form: $p \triangleq f(\sigma(t)) > 0$ for some $f : \mathbb{R}^m \to \mathbb{R}$. We write $(\sigma, t) \models \varphi$ to indicate that φ holds for $\sigma(t)$. The satisfaction of a signal σ at time t for any of the building block formulae in (2.1) is defined in the obvious way, except perhaps the one with the "until" operator **U**, which is given as:

$$(\sigma, t) \models \varphi \mathbf{U}_{[a,b]} \psi \Leftrightarrow \exists \tau \in [t+a, t+b].(\sigma, \tau) \models \psi \wedge \forall \tau' \in [t, \tau].(\sigma, \tau') \models \varphi.$$

For convenience, we can define the "eventually" $\Diamond$ and "always" $\Box$ operators as $\Diamond_{[a,b]}\varphi \triangleq \texttt{True}\mathbf{U}_{[a,b]}\varphi$ and $\Box_{[a,b]}\varphi \triangleq \neg(\Diamond_{[a,b]}\neg\varphi)$ such that $(\sigma, t) \models \Diamond_{[a,b]}\varphi$ if and only if φ is satisfied at least once within a time window of length $b - a$, a time units from t while $(\sigma, t) \models \Box_{[a,b]}\varphi$ requires that φ should always be satisfied within that time window.

2.2 Random Convex Program

The reliability of the proposed approach is based on the theory of random convex programs (RCP). Let $P[K]$ denote a (minimization) optimization problem with a known objective function and constraint set K, and let Obj$[K]$ denote the optimal objective value of $P[K]$. A constraint k is a supporting constraint if Obj$[K\backslash\{k\}] <$ Obj$[K]$. The setup for an RCP is the following:

$$\begin{aligned} &\min J(\alpha) \\ &\text{s.t.} \alpha \in Q(\delta_i), \forall \delta_1, ..., \delta_N \text{ i.i.d samples of } \delta, \end{aligned}$$

where $\alpha \in \mathbb{R}^n$, $Q(\delta_i) \subseteq \mathbb{R}^n$ is a convex set determined by δ_i, and $J(\alpha)$ is convex. $\delta \in \Delta$ is a random variable in the space Δ and $\{\delta_i\}$ are independently identically distributed samples of δ. Each δ_i would pose a convex constraint on α. If we randomly draw N samples of δ, and denote it as $\omega \doteq \delta_{1:N} \in \Delta^N$, then let $Q(\omega) \doteq \bigcap_{i=1}^{N} Q(\delta_i)$, define

$$V^*(\omega) = \mathbb{P}\left\{\delta \in \Delta : \text{Obj}([Q(\omega), Q(\delta)]) > \text{Obj}[Q(\omega)]\right\}, \tag{2}$$

which is the probability that an additional sample added on top of ω would change the objective value of the original optimization with constraints determined by ω. [14] gives upper bound on $\mathbb{P}(V^*(\omega) \geq \epsilon)$ given $1 \geq \epsilon > 0$ for a randomly drawn sequence of samples ω. First, we recall the following relevant lemma from [14]:

Lemma 1. *When $N \geq \zeta$, $\mathbb{P}\left\{\omega \in \Delta^N : V^*(\omega) > \epsilon\right\} \leq \Phi(\epsilon, \zeta - 1, N) \leq \Phi(\epsilon, n, N)$, where ζ is the Helly's dimension denoting the maximum number of supporting constraints, which is bounded by $n + 1$.*

$$\Phi(\epsilon, k, N) = \sum_{j=0}^{k} \binom{N}{j} \epsilon^j (1-\epsilon)^{N-j}$$

is the cumulative distribution of a binomial random variable, that is, the probability of getting no more than k successes in N Bernoulli experiments with success probability ϵ.

This is Theorem 3.3 in [14], which shows that the result of the RCP is likely to be true for unseen δ drawn from the same distribution under large N and small n. We will revisit this lemma in Section 3.1 to prove probabilistic correctness of the proposed method.

3 Approach

Given a dynamical system Σ, a specification about the initial condition φ_0, a controller φ_c and a performance specification φ_p, the goal is to verify that φ_p is satisfied by all the runs of the system, with high probability, when the control input and initial condition satisfy φ_0 and φ_c. However, since the system is interacting with the environment, φ_c and φ_0 alone typically do not imply φ_p, i.e., the verification fails trivially assuming that the environment can choose arbitrary behaviors. Therefore, we look for an assumption of the environment φ_e that is correct (or at least correct with high probability) such that $\Sigma \wedge \varphi_0 \wedge \varphi_e \wedge \varphi_c \Rightarrow \varphi_p$, where the implication is understood as

$$\forall (x(t), u(t), d(t)), ((x(t), u(t), d(t)) \models \Sigma \wedge \varphi_0 \wedge \varphi_c \wedge \varphi_e) \Rightarrow ((x(t), u(t), d(t)) \models \varphi_p).$$

We propose a framework that learns φ_e by using a falsification procedure in the loop, in addition to data of interaction between the system and the environment. Figure 1 depicts an overview of the framework being proposed. The falsification module takes a fixed controller, the reactive bound of the environment, and the specification φ_p as inputs, and either returns traces of the system evolution and the environment behavior that falsify φ_p under the given controller, or returns a flag saying that no falsifying trace could be found.

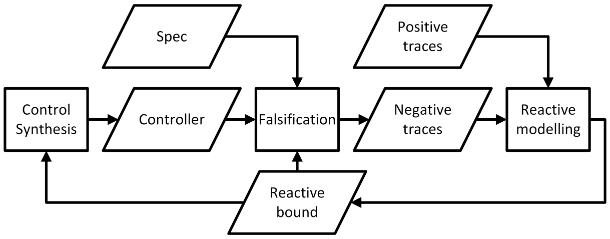


Figure 1: Verification with reactive modelling of the environment

In particular, we use the tool S-Taliro as an oracle falsifier, which uses stochastic sampling and can handle STL formulae, see [15] for details. When the falsification returns no trace, the procedure terminates and the verification is successful. When falsifying traces are found, they are fed into the reactive modelling module where the positive traces collected from the actual interaction with the environment and the negative traces from the falsification module go through a classification process and the output is a reactive bound that maps the agent's state x to a set of possible behaviors d for the environment, denoted as $S_d(x)$. To obtain a bound on the influence of x on the environment response d, we want to find a function $h : \mathcal{X} \times \mathcal{D} \to \mathbb{R}$ such that $h(x, d) \geq 0$ indicates that d is possible under x, and $h(x, d) < 0$ indicates that d is not possible under x. The set of possible behaviors for the environment given by the reactive bound can be represented as

$$S_d(x) = \{d \mid h(x, d) \geq 0\}. \tag{3}$$

Remark 1. Not every snapshot from the falsifying trace is included in the negative data. We use an ad-hoc selection scheme to pick out 'important' snapshots based on criterion such as the distance between the robots (Section 4.1), and lateral position for the AV (Section 4.2).

Remark 2. Since the data is in the format of snapshots, φ_e is limited to the form $\Box\varphi$ where φ has no dependence on time. One could use a parameterized form for φ_e and project the traces to the parameter space, such as in [8], but this limits φ_e to having only monotonic atoms.

3.1 $\mathcal{L}_1$ piecewise SVM for reactive modelling

Given the positive and negative traces, we need to learn an indicator function h, which then gives rise to the reactive bound. This can be solved as a classification problem. There are numerous classification tools in the literature, such as neural networks and logistic regression. For the reactive modelling problem, in addition to good classification accuracy, the following two requirements are critical: 1) the probability of false negative should be low, even for the unseen data, and 2) the classification result h should have an analytic form for its classification boundary. For the first requirement, note that $h(x, d) < 0$ indicates that d will not happen under x, and the verification process will ignore such environment input under x. Therefore, the probability of false negative should be very low to guarantee the correctness of the reactive bound and consequently guaranteeing safety. The reliability analysis of the proposed approach is based on the theory of random convex program (RCP), which we discuss in detail in Section 3.2. For the bound in Lemma 1, we would like the number of parameters for the classifier to be small – this prevents overfitting, and enables us to provide better probabilistic guarantees. The second requirement comes from the fact that the reactive bound will be used explicitly during verification and control synthesis. Therefore, its explicit form should be known.

Due to the two requirements, we choose Support Vector Machines (SVMs) with explicit features as the classification method. SVMs fit into the setup of Lemma 1 if the positive data is sampled from an i.i.d. distribution. In particular, we propose an novel expressive $\mathcal{L}_1$ piecewise SVM, which is based on the work on $\mathcal{L}_1$ SVM in [16]. [16] showed that with a proper cost function, the following optimization solves the $\mathcal{L}_1$ SVM:

$$\begin{aligned} &\min_{v,d,M} k^\intercal M \\ &\quad \text{s.t. } \|v\| \leq 1,\ y_i = 1 \Rightarrow M_i \geq 0, \\ &\qquad \forall i = 1, ..., N,\ v^\intercal \phi(z_i) - M_i + c = 0, \end{aligned} \tag{4}$$

where $z_i \in \mathbb{R}^m$ is the i-th data point and $\phi : \mathbb{R}^m \to \mathbb{R}^p$ maps the data to the feature space, the flag $y_i = 1$ for positive data points and $y_i = -1$ for negative data points. $v \in \mathbb{R}^p$ is the support vector and c is the offset, therefore the Helly dimension is $p + 2$. $M \in \mathbb{R}^N$ is the slack vector and k is the cost vector with $k_i > 0$ for all i. See section 6.1 for more detail. It is required that all the positive data points are correctly classified (no false negatives), which is needed for the reliability proof.

However, the $\mathcal{L}_1$ SVM suffers from the lack of expressibility, especially for high dimensional d. We propose two improvements on the SVM: 1) Algorithm 1, which generates multiple separating hyperplanes and represents the positive data region as a polytope. 2) Based on Algorithm 1, we introduce Algorithm 2, which allows a piecewise structure for the SVM where a different polytope represents the positive data in each region, and automatically synthesises the piecewise regions.

The original SVM generates one separating hyperplane in the feature space, which results in a reactive bound with a smooth boundary in $\mathcal{X} \times \mathcal{D}$. In order to make the reactive bound more expressive, we propose a piecewise $\mathcal{L}_1$ SVM with multiple separating hyperplanes. SVM with multiple separating hyperplanes is achieved with the following greedy algorithm:

Algorithm 1 $\mathcal{L}_1$ SVM with multiple separating hyperplanes

```
Input: positive data $\phi_p$, negative data $\phi_n$
$\phi_n^{active} \leftarrow \phi_n$
for i=1:$N_h$ do
    Perform $\mathcal{L}_1$ SVM with $\phi_p$ and $\phi_n^{active}$, get $v^i$, $c^i$, slacks $M_p$, $M_n$
    $\phi_n^{active} = \left\{\phi_n^{active}(j) | M_n(j) \geq \epsilon\right\}$
end for
```

N_h is the number of separating hyperplanes, ϕ_n^{active} is the set of negative data points that are close to the farthest separating hyperplane, ϵ is the threshold for picking ϕ_n^{active} and M_p and M_n are the slacks for positive and negative data. Each SVM computation generates one hyperplane with v^i and c^i and the indicator function is $h(z) = \min\limits_{i=1,.,N_h} \left\{(v^i)^\intercal z + c^i\right\}$. We can further improve the expressibility by introducing a piecewise structure , which is particularly helpful when the problem itself has a piecewise structure, as demonstrated in Section 4.1. Moreover, we develop an auto-tuning piecewise SVM that adjusts the dividing point automatically based on the data by the use of membership functions.

For clarity, we present the piecewise SVM with 2 regions, but note that it can be easily extended to cases with more than 2 regions. Let $g : \mathbb{R}^n \to \mathbb{R}$ be a scalar function and κ be a scalar variable. We will divide the state space by the threshold $g(z) = \kappa$. First, define the membership functions using the sigmoid:

$$m_1(z,\kappa) = \frac{1}{1+\exp(\gamma(g(z)-\kappa))},\ m_2(z,\kappa) = \frac{\exp(\gamma(g(z)-\kappa))}{1+\exp(\gamma(g(z)-\kappa))},$$

where γ is a tuning parameter that controls the steepness of the sigmoid. Note that $m_1(z,\kappa) + m_2(z,\kappa) = 1$. When there are $d > 2$ regions, one simply construct d membership functions that are nonnegative and add up to 1. With 2 regions, the original feature is extended to $\bar{\phi}(z) = [m_1(z,\kappa)\cdot\phi(z); m_2(z,\kappa)\cdot\phi(z)]$. We then perform $\mathcal{L}_1$ SVM with this new feature vector.

Once the SVM is trained, notice that by (4), $M_i = v^\intercal\bar{\phi}(z_i)y_i + dy_i$, taking derivative of the objective function over κ, we have:

$$\frac{d(k^\intercal M)}{d\kappa} = \sum\nolimits_i k_i y_i \frac{d(v^\intercal\bar{\phi}(z_i))}{d\kappa} = \sum\nolimits_i k_i y_i \left({v_{1:p}}^\intercal\phi(z_i)\frac{\partial m_1}{\partial\kappa} + {v_{p+1:2p}}^\intercal\phi(z_i)\frac{\partial m_2}{\partial\kappa}\right),$$

$$\left.\frac{\partial m_1}{\partial\kappa}\right|_z = -\left.\frac{\partial m_2}{\partial\kappa}\right|_z = \frac{\gamma m_1(z,\kappa)}{1+\exp(\gamma(g(z)-\kappa))},$$

so we obtain the analytic form of the gradient of the objective function over κ. The overall algorithm alternates between the $\mathcal{L}_1$ SVM and optimizing over κ by gradient descent, as in Algorithm 2.

Remark 3. The setup for $\mathcal{L}_1$ SVM allows for piecewise cost function of M by spliting $M = M^+ + M^-$ with $M^+ \geq 0, M^- \leq 0$, see [16] for detail. In the gradient descent step, we maintain the constraint by assigning a large penalty on M^-.

Algorithm 2 Auto-tuning piecewise $\mathcal{L}_1$ SVM

```
Initialize $\kappa$
for iter=1:T do
    Compute membership functions $m_1$, $m_2$
    Perform Algorithm 1 with $\bar{\phi}(z) = [m_1(z) \cdot \phi(z); m_2(z) \cdot \phi(z)]$
    Perform gradient descent to optimize $\kappa$
end for
```

3.2 Reliability analysis with RCP

Next, we provide reliability analysis for the reactive bound. For the ordinary $\mathcal{L}_1$ SVM, we have:

Theorem 1. *Given a positive data set with N points drawn i.i.d. from a fixed (not necessarily known) distribution, and a negative data set, let p be the dimension of the feature vector $\phi(z)$, $p+2 < N$, the $\mathcal{L}_1$ SVM in (4) is always feasible. Denote the solution as $[v, c]$, which satisfies $v^\intercal \phi(z_i) + d \geq 0,\ i = 1, 2, ..., N$. Then for an unseen data point z_{N+1} from the same distribution, for $0 < \epsilon < 1$, we have:*

$$\mathbb{P}\left\{\mathbb{P}\left\{\neg(v^\intercal \phi(z_{N+1}) + c > 0)\right\} > \epsilon\right\} \leq \sum\nolimits_{j=0}^{p+2} \epsilon^j (1-\epsilon)^{N-j}. \tag{5}$$

See Appendix 6.3 for proof. Theorem 1 gives an upper bound for the probability of the probability of misclassification for unseen data to be higher than a threshold, which decreases with the size of the dataset N and increases with the feature dimension p. For SVM with N_h hyperplanes, we provide the following corollary.

Corollary 1. *Given the condition in Theorem 1, the $\mathcal{L}_1$ SVM with N_h separating hyperplanes is always feasible and for $0 < \epsilon < 1$,*

$$\mathbb{P}\left\{\mathbb{P}\left\{\neg\left(\bigwedge_{i=1}^{N_h} (v^i)^\intercal \phi(z_{N+1}) + c^i > 0\right)\right\} > \epsilon\right\} \leq \sum\nolimits_{j=0}^{p+2} \left(\frac{\epsilon}{N_h}\right)^j \left(1 - \frac{\epsilon}{N_h}\right)^{N-j}$$

See Appendix 6.4 for proof.

Remark 4. The auto-tuning piecewise SVM in Algorithm 2 changes the optimization problem every time it updates κ, which does not allow us to directly apply Theorem 1. To overcome this, a simple solution is to separate the positive data points into two batches, using the first batch to find a good separation of the state space, i.e., find a good κ, and the second batch to obtain the reactive bound while fixing κ. The size of the second batch determines the probability of misclassification.

4 Case Study

4.1 Multi-robot navigation

As a toy example, we consider a multi-robot navigation problem consisting of two robots as shown in Fig. 2. We denote the positions of the two robots by $p_1, p_2 \in [-l, l]^2 \subset \mathbb{R}^2$. The robots are characterized by the integrator dynamics $\dot{p}_i = v_i$, where $i \in \{1, 2\}$ and v_i satisfies $\|v_i\|_2 \leq v^{\max}$. The specification for the system is to always maintain distance i.e., it has to satisfy the speficiation $\square_{[0,T]}\texttt{connected}$, where T is the time horizon for the STL specification and `connected` is a predicate defined by: $\texttt{connected} \triangleq \|p_1 - p_2\| \leq r^{\max}$. Here $r^{\max}$ can be thought of as the maximum communication range. The red robot R_1 is the controlled robot and it simultaneously pursues two objects, a moving target T_1 (with bounded velocity), and the blue robot R_2. It follows a given controller, which in simulation is set to be:

$$v_1 = \mathbf{sat}_{v^{\max}}(k_1(p_T - p_1) + k_2(p_2 - p_1)), \tag{6}$$

with gains k_1 and k_2, where p_T denotes the position of the target and $\mathbf{sat}_a(x) = x$ if $\|x\| \leq a$ and $a\frac{x}{\|x\|}$ otherwise. The motion of the blue robot R_2 follows a "black box" controller, and we would like to learn an over-approximation of its possible behavior as a function of the state. In particular, we pick a controller with a piecewise structure depending on the location R_2:

$$v_2 = \mathbf{sat}_{v^{\max}}\left(\begin{bmatrix} -0.4 & -\beta \\ \beta & -0.4 \end{bmatrix}(p_2 - p_1) + \Delta v_2\right), \tag{7}$$

where Δv_2 is a bounded random noise and $\beta = 1$ if $p_{2,1}(t) \leq 0$ and -1 otherwise ($p_{2,1}$ denote the X coordinate of R_2). The above controller roughly makes R_2 spiral counter-clockwise towards R_1 on the left half plane, and spiral clockwise on the right.

To initiate the process, we first collect data with simulation by enforcing (6) and (7) on R_1 and R_2 and let T_1 move randomly in the state space. The positive data collected consists of tuples of $[p_1, p_2, v_2]$, which contains information about how R_2 moves under different states. Recall (3), in the two robot case, the state x is $[p_1, p_2]$, and the environment input d is $[v_2, v_T]$, where $v_T = \dot{p}_T$ is the velocity of the target, but we do not explicitly learn a reactive model of v_T and the only constraint for which is the norm bound. In the falsification process, the falsifier can choose v_2 and v_T while v_1 follows (6). When no reactive bound is in place, the only constraint for v_2 is the norm bound $v^{\max}$, and the falsifier can easily find falsifying traces. The falsifying traces then generate negative data points with the same structure as the positive data, which is then fed to the reactive modelling module. The reactive modelling module utilizes the auto-tuning piecewise $\mathcal{L}_1$ SVM algorithm introduced in 3.1 with 3 separating hyperplanes and $g(x) = p_{2,1}$ to construct a reactive bound. We choose features that are linear in v_2 so that the resulting reactive bound is a polytope $S_{v_2}([p_1, p_2]) \in \mathcal{R}^2$, given $[p_1, p_2]$. The reactive bound is then fed to the falsifier, which would project the raw input of v_2 to $S_{v_2}([p_1, p_2])$. Since we construct the features for the SVM such that S_{v_2} is a polytope, the projection is easily solved with quadratic programming.

After 5 iterations of updating the reactive bound, the falsifier cannot find a falsifying trace, which means that the controller is verified under the learned reactive bound. Moreover, it turns out that the threshold κ converges to 3×10^{-3}, which is very close to the actual threshold at $\kappa = 0$.

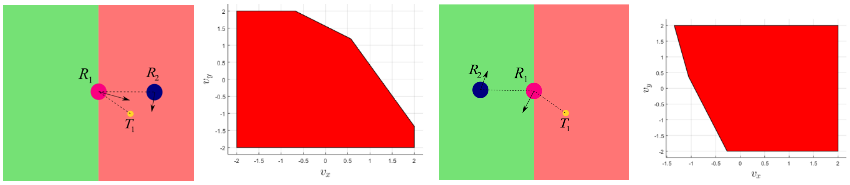


Figure 2: Two robot scenario and Reactive bounds.

Fig. 2 shows two robot positions with corresponding reactive bounds. When R_2 is on the right, the reactive bound allows it to spiral clockwise, while the direction of spiralling flips on the left side. But importantly, the worst-case v_2, which is to move away from R_1 with $v^{\max}$ is not allowed in both cases. For this example, we use 40000 snapshots with 45 features for the SVM. For $\epsilon = 0.01$, $\mathbb{P}(\mathbb{P}(\textit{misclassification}) \geq \epsilon) < 0.007$ by Corollary 1.

4.2 Lane Change

A practical application of the proposed method is verification of the lane change control for autonomous driving. We would like to guarantee with high probability that a given controller can safely finish a lane change within a given horizon. We consider a scenario as depicted in Fig. 3, where the autonomous vehicle (AV) attempts to make a lane change with the human driven vehicle (HV) on the back. The state of the system is $x = [\Delta X \quad \Delta Y \quad \Delta v \quad \psi]^\intercal$, where ΔX and ΔY are the longitudinal and lateral coordinate differences between the two vehicles, Δv is the velocity difference and ψ is the heading angle of the AV. The input of AV are the acceration a_1 and yaw rate r_1, and the input of the HV is the acceleration a_2. The dynamics is given by:

$$\dot{x} = [\Delta v \quad v_1 sin(\psi) \quad a_1 - a_2 \quad r_1]^\intercal . \qquad (8)$$

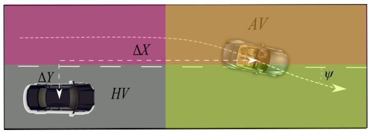


Figure 3: Lane Change Scenario

The specification for the problem is to always not collide and keep within the lane, and eventually finish the lane change within horizon T. Formally, the specification is expressed in STL:

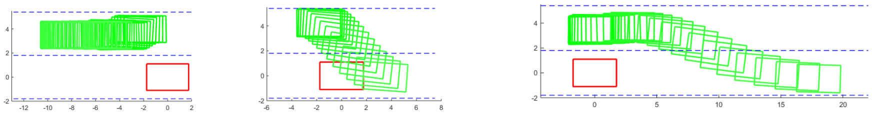
$$\square_{[0,T]}(\neg \mathbf{COL} \wedge \mathbf{LK}) \wedge \Diamond_{[0,T]} \mathbf{LC}, \qquad (9)$$

where $\mathbf{COL}$ stands for collision, $\mathbf{LC}$ stands for lane change and $\mathbf{LK}$ stands for lane keeping, which all can be represented as subsets of the state space:

$$\begin{aligned} \mathbf{COL} \Leftrightarrow & \quad |\Delta Y| \leq a \wedge |\Delta Y| \leq b \\ \mathbf{LC} \Leftrightarrow & \quad |\Delta Y| \leq \epsilon, \\ \mathbf{LK} \Leftrightarrow & \quad 0.5w - 0.5b \geq \Delta Y \geq -1.5w - 0.5b, \end{aligned} \qquad (10)$$

where a, b are the length and width of a typical car, w is the width of a lane and $\epsilon \in \mathbb{R}_+$ is a small constant. As an example, we consider a model predictive control scheme with mixed integer programming as the controller for the AV. As shown in Fig. 3, the AV should stay within the union of the two colored regions within the prediction horizon T, which is enforced by the "big M" procedure as a mixed integer linear constraint. The MPC controller takes the current value of a_2 and assumes an exponential decay within the prediction horizon $a_2(t) = a_2(0)e^{-t/\tau}$, which of course is not accurate, but only a prediction of the future. The lane keeping constraint is also enforced as a linear constraint and the objective function penalizes ΔY, driving the vehicle to finish the lane change.

The lane change problem was studied in [16], and we use the same source for positive data, which is from the safety pilot model deployment (SPMD) database with more than 50 million miles of naturalistic driving data [17]. The feature structure is also inherited from [16]. Following the procedure shown in Fig. 1, the falsification tool starts with simply the physical limit of a_2 and tries to falsify the specification in (9), the falsifying traces are then broken into snapshots and treated as negative data. The SVM procedure then generates the reactive bound for a_2. In the lane change case, it is not difficult to see that the the safety specification is monotonic w.r.t. a_2, i.e., it is always safer for the HV to decelerate. Therefore, the reactive bound for a_2 is in the form of an upper bound $a^{\star}_{\max}(x)$ that changes with the state x.



(a) Falsify by blocking (b) Falsify by collision (c) Success run

Figure 4: Verification of lane change

The result of verification for the MPC controller is shown in Fig. 4. Without the reactive bound, the falsification procedure is able to falsify the specification by accelerating and blocking the AV from finishing a lane change, as shown in Fig. 4a, and the verification procedure terminates after a maximum iteration number. After 4 iterations, the SVM presented in Section 3.1 generates a reactive bound that makes the falsification infeasible, i.e., verifies that the MPC controller satisfies the specification and a success run is shown in Fig. 4c. However, when we remove the collision avoidance constraint in the MPC controller, the falsification tool finds a falsifying trace by causing a collision with the AV (Fig. 4b), thereby providing feedback for the controller design process.

Check https://github.com/chenyx09/Reactive-modelling for the code implementing the proposed method with the two examples.
Run-time considerations The mean run times over 5 runs are 4977s and 29.6s for the robot problem and for the lane change problem respectively.

5 Conclusion

This paper presents a framework that combines falsification and specification learning to learn an over-approximation of the reactive behavior of the environment from real data. There are two key parts of the algorithm, the falsifier and the reactive modelling module. The falsifier can handle specifications written in temporal logic and generate falsifying traces, which is then used by the reactive modelling module together with the positive data to generate the reactive bound. The reliability of the reactive bound is guaranteed by the theory of RCP, which can give probabilistic guarantees determined by the amount of data available. We showed the capability of the proposed framework to handle environment behavior with a piecewise structure as well as demonstrating the result on a practical problem in autonomous driving with real-world human-driving data. The framework presented here provides a general approach for the diagnosis of autonomous agents interacting with complex environments, such as in the case of human-robot interaction.

References

[1] C. Fan, B. Qi, S. Mitra, and M. Viswanathan. DRYVR: data-driven verification and compositional reasoning for automotive systems. *CoRR*, abs/1702.06902, 2017. URL `http:`

//arxiv.org/abs/1702.06902.

[2] A. Kozarev, J. Quindlen, J. How, and U. Topcu. Case studies in data-driven verification of dynamical systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, pages 81–86, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3955-1. doi:10.1145/2883817.2883846. URL http://doi.acm.org/10.1145/2883817.2883846.

[3] C. Fan, B. Qi, and S. Mitra. Data-driven formal reasoning and their applications in safety analysis of vehicle autonomy features. *IEEE Design Test*, 35(3):31–38, June 2018. ISSN 2168-2356. doi:10.1109/MDAT.2018.2799804.

[4] A. Balkan, P. Tabuada, J. V. Deshmukh, X. Jin, and J. Kapinski. Underminer: A framework for automatically identifying nonconverging behaviors in black-box system models. *ACM Trans. Embed. Comput. Syst.*, 17(1):20:1–20:28, Dec. 2017. ISSN 1539-9087. doi:10.1145/3122787. URL http://doi.acm.org/10.1145/3122787.

[5] S. Haesaert, P. M. J. V. den Hof, and A. Abate. Data-driven and model-based verification: a bayesian identification approach. *CoRR*, abs/1509.03347, 2015. URL http://arxiv.org/abs/1509.03347.

[6] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(11):1704–1717, Nov 2015. ISSN 0278-0070. doi:10.1109/TCAD.2015.2421907.

[7] M. Vazquez-Chanlatte, J. V. Deshmukh, X. Jin, and S. A. Seshia. Logical clustering and learning for time-series data. In *International Conference on Computer Aided Verification*, pages 305–325. Springer, 2017.

[8] M. Vazquez-Chanlatte, S. Jha, A. Tiwari, M. K. Ho, and S. Seshia. Learning task specifications from demonstrations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5368–5378. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7782-learning-task-specifications-from-demonstrations.pdf.

[9] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: the case for cooperation. In *2013 IEEE International Conference on Robotics and Automation*, pages 2153–2160, May 2013. doi:10.1109/ICRA.2013.6630866.

[10] M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646, May 2015. doi:10.1109/ICRA.2015.7139555.

[11] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2016. doi:10.15607/RSS.2016.XII.029.

[12] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, and S. S. Srinivasa. Learning the communication of intent prior to physical collaboration. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 968–973, Sep. 2012. doi:10.1109/ROMAN.2012.6343875.

[13] A. Donzé. On signal temporal logic. In *International Conference on Runtime Verification*, pages 382–383. Springer, 2013.

[14] G. C. Calafiore. Random convex programs. *SIAM Journal on Optimization*, 20(6):3427–3464, 2010.

[15] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[16] Y. Chen, N. Sohani, and H. Peng. Modelling of uncertain reactive human driving behavior: a classification approach. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3615–3621. IEEE, 2018.

[17] D. Bezzina and J. Sayer. Safety pilot model deployment: Test conductor team report, usdot report no. dot hs 812 171, 2015.

6 Appendix

6.1 A simple derivation for $\mathcal{L}_1$ SVM

Given a data set of features together with their labels $(\phi(z_i), y_i)_{i=1}^N$. Let's say we want to find a hyperplane $\mathcal{H}$ characterized by a normal vector w and an offset c to classify them such that

1. all positive data points lie on one side of $\mathcal{H}$
2. all positive data points lie as close to the hyperplane as possible
3. negative data points lie as far into or close to the other side of $\mathcal{H}$ as possible

For each $i \in \{1, 2, \dots, N\}$, the distance of $\phi(z_i)$ to the hyperplane $\mathcal{H}$ is equal to $|M_i|$ where M_i is the (unique) solution to the equation

$$w^T \left(\phi(z_i) - M_i \frac{w}{\|w\|} \right) + c = 0$$

Then, requiring that all positive data points must all lie on one side of the hyperplane like in requirement 1 is equivalent to adding the constraint

$$y_i = 1 \Rightarrow M_i \geq 0 \tag{11}$$

Actually (11) says that all positive data points should lie on the positive side of the hyperplane, but by symmetry this is not a loss of generality. Consider a linear objective (penalty) function of the form $k^\intercal M$ where $M \in \mathbb{R}^N$ has M_i as its i-th component. Then requirements 1, 2, and 3 translate to a problem of minimizing $k^\intercal M$ with the weight k satisfying $k \succ 0$ (element-wise positive) subject to (11).

- For each i such that $y_i = 1$ (and hence $M_i \geq 0$ by requirement 1 and (11)), k_i is required to be strictly positive to penalize $\mathcal{H}$ being far away from $\phi(z_i)$ (requirement 2).
- For each i such that $y_i = -1$ (negative data points), having $k_i > 0$ will "force" M_i to be small (i.e., closer to $-\infty$ on the real axis) to minimize the penalty function and in the process, satisfying requirement 3.

A convex relaxation of the above optimization problem is

$$\begin{aligned} \min_{v,d,M} \; & k^\intercal M \\ \text{s.t. } & \|v\| \leq 1 \\ & v^\intercal \phi(z_i) - M_i + d = 0 \\ & y_i = 1 \Rightarrow M_i \geq 0 \end{aligned} \tag{12}$$

The two optimization problems are equivalent if the optimal solution is negative. Optionally, if we want to weigh correctly or wrongly classified negative data points differently, we can modify the optimization problem to.

$$\begin{aligned} \min_{v,d,M} \; & k_p^\intercal M_p + k_{nc}^\intercal M_{nc} + k_{nw}^\intercal M_{nw} \\ \text{s.t. } & \|v\| \leq 1 \\ & v^\intercal \phi(z_i) - M_i + d = 0 \\ & y_i = 1 \Rightarrow M_i \geq 0 \\ & y_i = -1 \Rightarrow M_i = M_i^{nw} + M_i^{nc} \wedge M_i^{nw} \geq 0 \wedge M_i^{nc} \leq 0 \end{aligned} \tag{13}$$

where $k_p, k_{nw}, k_{nc} \succeq 0$ and $k_{nc} \preceq k_{nw}$.

6.2 Helly's dimension for a variation of RCP

The following theorem is used in the proof of Theorem 1.

Theorem 2. *For the following RCP:*

$$\begin{aligned} &\min_{\alpha,\beta} J\left(\alpha,\beta\right) \\ &\text{s.t.} f\left(\alpha,\delta_i\right) \le 0, \forall \delta_1, ..., \delta_N \ i.i.d., \\ &\quad g\left(\alpha,\beta\right) \le 0. \end{aligned} \tag{14}$$

where $\alpha \in \mathbb{R}^{n_1}$, $\beta \in \mathbb{R}^{n_2}$, *$f$, g and J convex, the Helly's dimension* $\zeta \le n_1 + 1$.

Proof. Since J is convex, define

$$\bar{J}\left(\alpha\right) = \begin{cases} \min\limits_{\beta} \ J\left(\alpha,\beta\right) \ \text{s.t. g}\left(\alpha,\beta\right) \le 0, & \text{if } \exists \beta, \text{g}\left(\alpha,\beta\right) \le 0 \\ \infty, \quad \text{otherwise} \end{cases} \tag{15}$$

Then the RCP in (14) is equivalent to

$$\begin{aligned} &\min_{\alpha} \bar{J}\left(\alpha\right) \\ &\text{s.t.} f\left(\alpha,\delta_i\right) \le 0, \forall \delta_1, ..., \delta_N \ i.i.d. \end{aligned} \tag{16}$$

When $\bar{J} \ne \infty$, the number of supporting constraint is at most $n_1 + 1$; when $\bar{J} = \infty$, the number of supporting constraint is zero. This proves that $\zeta \le n_1 + 1$. □

6.3 Proof of Theorem 1

Proof. Feasibility can be seen by noticing that the problem is convex and $[w, c] = 0$ is a solution. Then note that $[w, c]$ is the optimization variable, and by Theorem 2, Helly's dimension is upper bounded by $p+2$, the upper bound on the probability of misclassification for unseen data is obtained by directly using theorem 3.3 in [14], see the proof therein. □

6.4 Proof of Corollary 1

Proof. The multi-hyperplane SVM can be viewed as the conjunction of N_h SVMs, therefore we have

$$\mathbb{P}(\bigvee_{i=1}^{N_h} ((w^i)^\intercal \phi(z_{N+1}) + c^i < 0)) \le \sum_{i=1}^{N_h} \mathbb{P}((w^i)^\intercal \phi(z_{N+1}) + c^i < 0), \tag{17}$$

where for each probability on the right, pick $\epsilon' = \frac{\epsilon}{N_h}$ and apply Theorem 1. Then sum up the upper bounds to reach the conclusion. □