Piece-Wise Relational Models for Falsification of Safety Properties in Hybrid Systems.

Aditya Zutshi[1], Sriram Sankaranarayanan[1], Sergio Mover[1] and Jyotirmoy V. Deshmukh[2]

Abstract— **In this work we address the problem of finding unsafe behaviors with given a safety property for a hybrid dynamical system. We treat the system as a black box and approach the problem in two distinct steps. First, we model the behavior of the system using a piecewise affine discrete time model. Such a model is incomplete in nature and is computed with respect to the given property. Next, we encode the search for falsification as a bounded model checking query and use an SMT solver to find a counter example. If found, we check the existance of the violation in the original system to ensure reproducibility.**

I. Introduction

Model based design has been quite successful in the industry. Tools like Simulink®are often used to create models describing hybrid dynamical systems. Testing such systems for safety properties remains a hard problem, and the current approaches consist of either guided testing using numerical simulations or translating to hybrid automata. We now suggest an approach which lies in the middle.

In this work we propose a two step process for searching violations of the properties. This involves (a) learning a piece-wise affine model and (b) exhaustively searching it for violations. It must be noted that the violation found in the model might not be valid in the original system, depending on the accuracy of the model.

The piece-wise affine model is the aggregation of affine maps, each approximating observed local behavior of the system. The model is discrete time and 'relational'. It relates the states reachable in a fixed time step Δ to a given state.

In our previous work [**?**], we analyzed systems by restricting ourselves to their black box semantics. This enabled us to reason about the state-space reachability of the system without a direct analysis of its structure. Using a coarse abstraction which we searched on-the-fly, we could observe the local dynamics as required. This gave us an efficient procedure to find abstract counter-examples. However, to concretize the counter-examples, a 'closer look' or refinement of the abstraction is required. The grid based state-space abstraction was refined by splitting the relevant cells (abstract states) into smaller ones. As noted previously, due to the curse of dimensionality, such a uniform splitting is an expensive operation, and can not scale to higher dimensions.

We further our exploration of trajectory segment based methods; and explore an alternative approach to overcome the explosion in abstract states. Instead of selectively refining the abstraction, we compute a model of the black box system and use bounded model checking to find a concrete counter-example in the model [1]. We then use this counter-example to guide the search towards a counter-example in the original system.

More specifically, we use regression to quantitatively estimate the discovered relations, which were witnessed by trajectory segments, by affine maps. These maps approximate locally observed behaviors, which are incorporated into the sampled reachability graph as edge annotations. The resulting graph or the Piece-Wise Affine (PWA) relational model can be interpreted as an infinite state discrete transition system and model checked for time bounded safety properties. A counter-example if found, can indicate the presence of a violation in the system.

Towards the end of the chapter, we discuss extensions to data driven approaches (instead of simulator driven), where instead of a simulator, a fixed set of data is provided as the behavioral description of the system. Using a combination of relational modeling, and program analysis we outline the future work for falsifying properties of SDCS.

II. Overview

We now briefly describe the approach. Given a numerical simulator SIM and an initial abstraction defined by a quantization function Q_ϵ and a time step Δ, we use scatter-and-simulate (parameterized by number of samples n as described in Chapter **??**) to explore the abstract graph $\mathcal{H}(\Delta)$. The result is a graph G, which has a finite number of abstract states C (or cells) and edges (C, C') iff $C \overset{\Delta}{\rightsquigarrow} C'$.

Instead of using a CEGAR like loop (as in Chapter **??**), we use the generated trajectory segments to learn quantitative models describing the local behavior of the system. These models are defined by a set of relations $R \subseteq \mathcal{X} \times \mathcal{X}$ for each edge of the reachability graph.

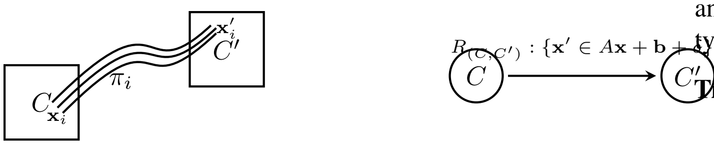
Recall that each edge (C, C') of the graph G denotes an observed trajectory segment between the respective cells. The graph abstraction G only states that there exists a state $\mathbf{x} \in C$ from which the system can evolve to a future state $\mathbf{x}' \in C'$. To increase the precision we iteratively refined the abstraction by state splitting. Instead, we now propose an 'enrichment' G^R of G by computing a set of local relations $R_{(C,C')}(\mathbf{x}, \mathbf{x}')$ for every edge (C, C'), which non-deterministically describe relations between $\mathbf{x} \in C$ and $\mathbf{x}' \in C'$. This is illustrated in Fig. 1 (compare with Fig. **??**).

*This work was not supported by any organization

[1]University of Colorado, Boulder {aditya.zutshi, srirams, sergio.mover}@colorado.edu

[2]Toyota Technical Center, LA jyotirmoy.deshmukh@toyota.com

[1]Due to modeling errors, this might not be reproducible in the original black-box system.


C
$\mathbf{x}_i$
π_i
C'
$\mathbf{x}'_i$
$R_{(C,C')} : \{\mathbf{x}' \in A\mathbf{x} + \mathbf{b} + \delta\}$
C
C'

(a) (b)

Fig. 1: (a) Trajectory segments π_i are used to compute the relation $R_{(C,C')}$ that annotates the edge in (b). $R_{(C,C')}$: $\{\mathbf{x}' \in A\mathbf{x} + \mathbf{b} + \delta\}$ is an interval affine relation defined by an affine map (matrix A and vector $\mathbf{b}$) and an error interval (vector of intervals δ).

The enriched graph G^R captures the underlying local forward dynamics describing the evolution of the system in each abstract state. We represent the dynamics using an affine model with an interval error. Such a model can either be approximated using learning methods or computed as a sound (over) approximation using reachability set computation methods. Because we assume black box semantics, we only present the former. Using regression analysis on the *start* and *end* states of the witnessed trajectory segments between two cells, we compute an approximate discrete map along with an error estimate. Moreover, using the simulation function SIM, additional trajectory segments (or data) can be generated if required. The data can be separated into a training set and testing set to compute the map and the error respectively. The latter case can be explored if the symbolic dynamics of the system are known. A tool like FLOW* [1] can be used to find the reachable set map.

Observe that G^R, a directed reachability graph, is rich enough to search for concrete behaviors in the system. We call it a time parameterized PWA relational abstraction. It can be interpreted as an infinite state discrete transition system, and we can use off-the-shelf bounded model checkers to find concrete violations of a given safety property, and even other temporal properties. We now discuss the background required to present our ideas.

III. Background

In this section, we present the basics of relational abstractions and fundamentals of regression analysis.

A. Relational Abstraction

For general hybrid dynamical systems, algorithmic ways for computing analytical solutions do not exist. Discretization (along with suitable abstractions) is often employed to transform the systems into a discrete transition system. Relational abstractions is one such idea, which abstracts continuous dynamics by discrete relations using appropriate *reachability invariants*. Both time independent and time dependent relations have been proposed; the former captures all reachable states over all time, whereas, the latter explicitly includes time by relating reachable states to time. Thus it can prove timing properties whereas, time independent relational abstractions can not. In both cases, the resulting abstractions can be interpreted as discrete transition systems and can be analyzed using model checkers. We briefly discuss these two types of abstractions.

Timeless Relational Abstraction

Relational abstractions for hybrid systems were proposed in [9]. For a hybrid automaton model, they can summarize the continuous dynamics of each mode using a binary relation over continuous states. The resulting relations are timeless (independent) and hence valid for all time as long as the mode invariant is satisfied. The relations take the general form of $R(\mathbf{x}, \mathbf{x}') \bowtie 0$, where $\bowtie$ represents one of the relational operators $=, \geq, \leq, <, >$. For example, an abstraction that captures the monotonicity with respect to time for the differential equation $\dot{x} = 2$ is $x' > x$. The abstraction capturing the relation between the set of ODEs: $\dot{x} = 2$, $\dot{y} = 5$, is $5(x' - x) = 2(y' - y)$.

Such relation summaries, discretize the continuous evolution of a given system into an infinite state transition system, in which safety properties can be verified using k-induction, or falsified using bounded model checkers. The relationalization procedure involves finding suitable invariants in a chosen abstract domain, such as affine abstractions, eigen abstractions or box abstractions [9]. For this, different techniques like template based invariant generation [2], [5] can be used.

Timed and Time-Aware Relational Abstractions

As the above discussed relations are timeless, we cannot reason over the timing properties of the original system. Moreover, they are not suitable for time-triggered systems. Timed relational abstractions [10] and time-aware relational abstractions [6] were proposed to overcome these shortcomings by explicitly including time in the relations. In addition, both can be more precise than their timeless counterpart, but unlike it, assume continuous affine dynamics.

To analyze time-triggered systems like an SDCS modeled by an affine hybrid automata [2], timed relational abstractions can be computed directly from the solutions of the underlying affine ODEs. Recall that the solution of an affine ODE $\dot{\mathbf{x}} = A\mathbf{x}$ can be represented using the matrix exponential as $\mathbf{x}(t) = e^{tA}\mathbf{x}(0)$. This gives us the timed relation $\mathbf{x}' = e^{tA}\mathbf{x}$. Clearly, the relation is non-linear with respect to time. However, for the case of SDCS with a fixed sampling time period, we obtain linear relations in $\mathbf{x}$. We demonstrated the usefulness of timed relational abstractions for the case of linear systems in [10], and now incorporate the idea to discretize black box dynamical systems.

Similar to timeless relational abstractions, time-aware relational abstractions [6] construct binary relations between the current state $\mathbf{x}$ of the system and any future reachable state $\mathbf{x}'$. The difference lies with the latter also constructing relations between the current time t and any future time t'. This is achieved by a case by case analysis of the eigen structure of the matrix A (of an affine ODE). Separate abstractions are used for the case of linear systems with constant rate, real eigen values and complex eigenvalues.

[2] An affine hybrid automata is restricted to the ODEs, resets and guards being affine.

B. Learning Dynamics using Simple Linear Regression

Regression Analysis

In statistics, regression is the problem of finding a *predictor*, which can suitably predict the relationship between the given set of observed input $\mathbf{x}$ and output $\mathbf{y}$ vectors. In other words, assuming that $\mathbf{y}$ depends on $\mathbf{x}$, regression strategies find either a parameterized or a non-parameterized prediction function to explain the dependence. We now discuss simple linear regression, which is parametric in nature and searches for an affine predictor. It is also called *ordinary least squares*.

Ordinary Least Squares (OLS)

Let the data set be comprised of N input and output pairs $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^n$ and $y \in \mathbb{R}$. If $N > n$, which is the case in the current context of *finding the best fit*, the problem is over-determined; there are more equations than unknowns. Hence, a single affine function cannot be found which satisfies the equation $\forall i \in \{1 \dots N\}. y_i = \mathbf{a}^T \mathbf{x}_i + b$. Instead, we need to find the 'best' choice for $\mathbf{a}$ and b. This is formally defined using a loss function. For the case of simple linear regression or OLS, the loss function is the sum of squares of the errors in prediction. The task is then to determine the vector of coefficients $\mathbf{a}$ and an offset constant b, such that the least square error of the affine predictor is minimized for the given data set.

$$\underset{\mathbf{a}, b}{\operatorname{argmin}} \sum_{i=1}^{N} \left(\mathbf{y}_i - (\mathbf{a}^T \mathbf{x}_i + b)\right)^2 \tag{1}$$

To ease the presentation, we can rewrite the above as a homogeneous expression by augmenting $\mathbf{x}$ by $\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$ and replacing $\mathbf{a}$ and b by the vector $\mathbf{ab} = \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix}$. The equation 1 now becomes

$$\underset{\mathbf{ab}}{\operatorname{argmin}} \sum_{i=1}^{N} (\mathbf{y}_i - \mathbf{ab}^T \hat{\mathbf{x}}_i)^2$$

The solution of OLS can be analytically computed as

$$Ab = (X^T X)^{-1} X^T \mathbf{y}$$

where X is the matrix representing the horizontal stacking of all $\hat{\mathbf{x}}$. The details can be found in several texts on learning and statistics [4].

Given a time invariant dynamical system $\mathbf{x}' = \textsc{sim}(\mathbf{x}, \Delta)$, where $\mathbf{x} \in \mathbb{R}^n$ we can use OLS to approximate its trajectories at fixed time step Δ by a discrete map $\mathbf{x}' = A\mathbf{x} + \mathbf{b}$, where $A = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$. This map is a global relational model for the system. The data set for the regression is a set of pairs $\{(\mathbf{x}, \mathbf{x}') | \mathbf{x}' = \textsc{sim}(\mathbf{x}, \Delta)\}$ and can be generated on demand. Another set can be generated to compute the error δ as an interval of vectors, where each element is an interval $\delta_i \in [\delta_{min}, \delta_{max}]$.

Affine maps are poor approximations for arbitrary non-linear functions. Hence, we use a collection of affine maps to approximate the local behaviors (in state-space) of the system $\textsc{sim}$. This results in a piece wise approximation, as described in the next section.

C. Piecewise Affine (PWA) Transition System

We define the PWA transition system as a transition system (ref. Sec. **??**) $\rho : \langle L, \mathcal{V}, \mathcal{T}, l_0, \Theta \rangle$ where each transition $\tau \in \mathcal{T}$, is associated with a transition relation of the form

$$\rho_\tau(v, v') \subseteq \{(v, f_\tau(v)) \mid g_\tau(v) \wedge g'_\tau(f_\tau(v))\}$$

where f_τ is an affine map relating the states and g_τ, g'_τ are affine guards (pre and post conditions) on the states. The assertion on the initial states Θ is also affine.

We can now use a PWA transition system ρ to approximate the behavior of a hybrid dynamical system S defined by a simulator $\textsc{sim}_S$ [3]. Each transition of ρ represents a discrete time step Δ. Let the state-space of S be given by $\mathbf{x} \in \mathcal{X}^n$, then the states of ρ are also given by $\mathbf{x} \in \mathcal{X}^n$. The affine guard predicates g_τ are given by a conjunction of hyperplanes and hence can be represented in matrix form as a polyhedron (defined by m constraints) in the state-space

$$g : C\mathbf{x} - \mathbf{d} \leq 0$$

where C is an $m \times n$ matrix and $\mathbf{d}$ is a vector of length m. The affine maps f_τ can be represented in matrix form as $f : \mathbf{x} \mapsto A\mathbf{x} + \mathbf{b}$. To make f_τ 'richer', we incorporate an error term δ which is defined by a vector of intervals $[\delta_{\min}, \delta_{\max}]$. Hence,

$$f : \mathbf{x} \mapsto A\mathbf{x} + \mathbf{b} + \delta$$

defines a set valued relation of the form $R : \{(\mathbf{x}, \mathbf{x}') \mid \mathbf{x}' \in f_\tau(\mathbf{x})\}$.

We now formalize the PWA affine transition system for a dynamical system.

Definition 3.1 (PWA Transition System): Given a hybrid dynamical system over a state-space $\mathcal{X}^n$, a PWA transition system ρ is given by the tuple $\langle L, \mathbf{x}, \mathcal{T}, l_0, \Theta \rangle$, where $\tau \in \mathcal{T}$ are affine transition relations and Θ is an affine predicate over $\mathbf{x}$ and the states $\mathbf{x} \in \mathcal{X}^n$. The transition relation is then defined by $\mathcal{T}$ with n transition relations as follows

$$\mathcal{T} = \begin{cases} g_1(\mathbf{x}) \wedge g'_1(\mathbf{x}') \implies \mathbf{x}' \in f_1(\mathbf{x}) \\ \dots \\ g_n(\mathbf{x}) \wedge g'_n(\mathbf{x}') \implies \mathbf{x}' \in f_n(\mathbf{x}) \end{cases} \tag{2}$$

where $f_i : \mathbf{x} \mapsto A_i\mathbf{x} + \mathbf{b}_i + \delta_i$, $g_i(\mathbf{x}) : C_i\mathbf{x} - \mathbf{d}_i \leq 0$, $g'_i(\mathbf{x}') : C'_i\mathbf{x}' - \mathbf{d}'_i \leq 0$ and $\mathbf{x}'$ denotes the next state of the system.

A PWA model is *deterministic*, iff for every state $\mathbf{x} \in \mathcal{X}^n$, a unique guarded affine map is satisfied and all errors δ are singletons. However, in practice such a case rarely exists. A PWA model will be usually non-deterministic, both due to multiple choice of transition relations and the reachable states at the k^{th} time step (or after $k\Delta$ units of time) being a set. Abusing the notation, we denote the set of states reachable in a single step in ρ by $\mathbf{x}' = \rho(\mathbf{x})$.

[3] Due to f_τ being restricted to an affine form, we can only approximate general hybrid systems.

A PWA transition system is *complete*, iff for every state $\mathbf{x} \in \mathcal{X}^n$, there exists at least one satisfied transition relation $\mathcal{T}$. If this is not the case, the system can deadlock, with no further executions. This usually results from incorrect modeling of SDCS, and from here on, we do not consider such cases.

IV. Relational Modeling

We now describe how relational models can be computed for a given black box system by enriching the abstract reachability graph obtained by scatter-and-simulate. We first formalize the notion of an enriched graph, and then show how they can be interpreted as a PWA transition system using k-relational modeling.

A. Abstract Enriched Graph

The existential abstraction relation in Chapter **??** was defined as follows:

$$C \overset{t}{\rightsquigarrow} C' \iff \exists \mathbf{x} \in C.\ \exists \mathbf{x}' \in C'.\ \mathbf{x} \overset{t}{\rightsquigarrow} \mathbf{x}'$$

The abstract relation $\overset{t}{\rightsquigarrow}$ can be enriched by incorporating the affine relation between $\mathbf{x}$ and $\mathbf{x}'$. For an arbitrary dynamical system, such a relation can be rarely represented using an exact affine map. This is due to the presence of non-linear and hybrid behaviors. But, an affine map can always be estimated with an error.

If the system dynamics are completely specified in the form of a white box model, we can use first order approximations to find the affine expressions for the relations. Using a tool like FLOW*, we can obtain sound over-approximate affine maps of the form $A\mathbf{x} + [\mathbf{b}^l, \mathbf{b}^h]$, where $[\mathbf{b}^l, \mathbf{b}^h]$ denotes the interval of vectors, such that, every element b_i of the vector $\mathbf{b}$ is contained in the respective scalar interval $b_i \in [b_i^l, b_i^h]$.

For the case of black box systems, such a sound approximation is not possible. Instead, we rely on a statistical method like simple linear regression to estimate the affine map $f : A\mathbf{x} + \mathbf{b}$. We then estimate the error δ and generalize f to an interval affine map as before $f : A\mathbf{x} + \mathbf{b} + \delta$. Finally, we get an abstraction with the below relation

$$C \overset{f}{\rightsquigarrow} C' \iff \exists \mathbf{x} \in C.\ \exists \mathbf{x}' \in C'.\ \mathbf{x}' \in A\mathbf{x} + \mathbf{b} \pm \delta.$$

However, in the presence of complex non-linear behavior, using a single affine map can lead to a poor approximation. To increase the precision, one can use more than one affine map. Let us denote this set as $R_{(C,C')}$ (ref. Sec. II).

$$R_{(C,C')}(\mathbf{x}, \mathbf{x}') : \{f_i \mid \exists \mathbf{x} \in C.\ \exists \mathbf{x}' \in C'.\ \mathbf{x}' \in f_i(\mathbf{x})\}$$

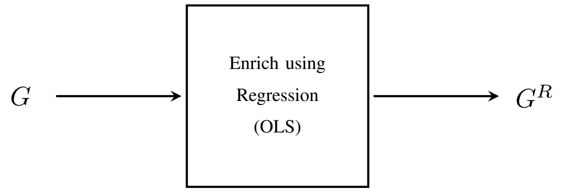


Fig. 2: Using OLS, G^R is computed by determining the appropriate $R_{(C,C')}$ for each edge of G.

As shown in Fig. 2, we use OLS to compute an enrichment of the abstraction graph G. For every edge $(C, C') \in edges(G)$, the set of relations $R_{(C,C')}(\mathbf{x}, \mathbf{x}')$ is computed. The semantics of the enriched edge are clearly non-deterministic. Form a state $\mathbf{x} \in C$, any $f_i \in R_{(C,C')}$ can be taken as long as $\mathbf{x}' \in C$' is satisfied. This interpretation results in an infinite state transition system. We now detail the construction of the set $R_{(C,C')}$ using k-relational modeling.

B. k - Relational Modeling

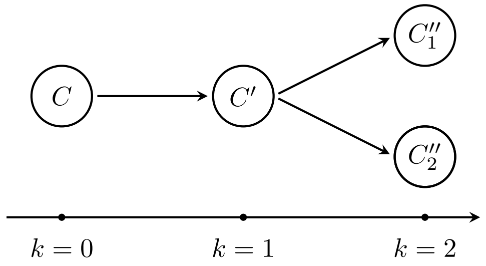


Fig. 3: Nodes/Cells of G shown along with increasing values of k.

Given a G, when computing the set of relations for an edge R, we consider only the set of abstract states reachable at a specific time step. Intuitively, one might consider the states reachable in one time step Δ. However, such a process can be generalized by looking at the states reachable at $0\Delta, 1\Delta, \ldots, k\Delta$ time steps.

Using Fig. 3, we illustrate this notion. To compute $R_{(C,C')}$, we split the data set $D(C, C')$ consisting of trajectory segments (of time lengths $k\Delta$) as follows. We observe the local behavior of the system by noting the evolution of the system S from a state $\mathbf{x}(t) \in C$ for a time length dependant on k. For

- $k = 0$: we observe all trajectory segments of length Δ.
- $k = 1$: we observe trajectory segments of length Δ, which satisfy $\mathbf{x}(t + \Delta) \in C'$.
- $k = 2$: we observe trajectory segments of length 2Δ, and split them in two sets (a) and (b) on the basis of the cells reached at time $(t + \Delta)$ and $(t + 2\Delta)$. (a)
  1) $\mathbf{x}(t + \Delta) \in C' \wedge \mathbf{x}(t + 2\Delta) \in C_1''$
  2) $\mathbf{x}(t + \Delta) \in C' \wedge \mathbf{x}(t + 2\Delta) \in C_2''$
- $k = n$: we observe trajectory segments of length $n\Delta$, and split them in to multiple sets on the basis of the cells reached at time $(t + \Delta)$, $(t + 2\Delta)$, $\ldots$, $(t + n\Delta)$.

k-relational modeling can be understood as a heuristic, which uses the underlying abstraction to differentiate behaviors of the system which 'diverge' (or are revealed to be 'distinct' at a future time). Such a heuristic can be useful in increasing the precision of the learnt affine maps. We now formalize this notion for $k = 0$, 1, and n and illustrate using examples.

0-Relational Model

When $k = 0$, the 0-relational model is a PWA transition system which only defines the evolution of states $\mathbf{x}$, and does not specify the reachable cell. The guard predicates of

$$\mathcal{T} \equiv g_1(\mathbf{x}) : \mathbf{x} \in C \implies A_1\mathbf{x} + \mathbf{b}_1 + \delta_1$$

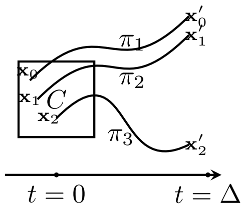


Fig. 4: All the trajectory segments will be used to construct the model; $\mathcal{D} = \{\pi_1, \pi_2, \pi_3\}$.

the relations are defined over cells: $g_i(\mathbf{x}) : \mathbf{x} \in C_i$ while $g'_i(\mathbf{x}') : True$.

We use regression to estimate the dynamics for the outgoing trajectory segments form a cell C. Hence, the data set $\mathcal{D}$ for the regression includes all trajectory segments beginning from the same cell

$$\mathcal{D} = \{\pi_t | start(\pi_t) \in C\}.$$

This includes trajectory segments ending in different cells, as shown in Fig. 4. For N cells, this results in a ρ with N transition relations.

$$\mathcal{T} = \begin{cases} g_1(\mathbf{x}) : \mathbf{x} \in C_1 \implies A_1\mathbf{x} + \mathbf{b}_1 + \delta_1 \\ \dots \\ g_n(\mathbf{x}) : \mathbf{x} \in C_n \implies A_n\mathbf{x} + \mathbf{b}_n + \delta_n \end{cases}$$

Note that this can be quite imprecise when the cells are big, containing regions of state-space with complex dynamics. This is true for both non-linear systems and hybrid dynamical system, where a cell can contain two or more modes with differing continuous dynamics.

1-Relational Model

$$\mathcal{T} = \begin{cases} g_1(\mathbf{x}) : \mathbf{x} \in C \wedge g'_1(\mathbf{x}) : \mathbf{x} \in C'_1 \implies A_1\mathbf{x} + \mathbf{b}_1 + \delta_1 \\ g_2(\mathbf{x}) : \mathbf{x} \in C \wedge g'_2(\mathbf{x}) : \mathbf{x} \in C'_2 \implies A_2\mathbf{x} + \mathbf{b}_2 + \delta_2 \end{cases}$$

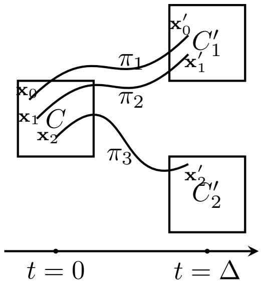


Fig. 5: The data gets split into two sets $\mathcal{D}_1 = \{\pi_1, \pi_2\}$ and $\mathcal{D}_2 = \{\pi_3\}$ and two relations: $R_{(C,C'_1)}$ and $R_{(C,C'_2)}$, each with one affine map, are constructed.

To improve the preciseness of learnt dynamics, we include the reachability relation in the regression. For every relation $C \overset{t}{\rightsquigarrow} C'$, the data set $\mathcal{D}$ is comprised only of trajectory segments π_t which start and end in the same set of cells.

$$\mathcal{D} = \{\pi_t | start(\pi_t) \in C \wedge end(\pi_t) \in C'\}.$$

For N edges in G, 1-relationalization results in a ρ with N transition relations.

Fig. 5 illustrates the $k = 1$ refinement of the case shown in Fig. 4. The data set $\mathcal{D}$ is split into two data sets $\mathcal{D}_1 = \{\pi_1, \pi_2\}$ and $\mathcal{D}_2 = \{\pi_3\}$, using which, two relations are constructed. The resulting 1-relational model is at least as precise as the corresponding 0-relational model.

k-Relational Model

$$\mathcal{T} = \begin{cases} g_1(\mathbf{x}) : \mathbf{x} \in C \wedge g'_1(\mathbf{x}) : \mathbf{x} \in C'_1 \implies A_1\mathbf{x} + \mathbf{b}_1 + \delta_1 \\ g_1(\mathbf{x}) : \mathbf{x} \in C \wedge g'_1(\mathbf{x}) : \mathbf{x} \in C'_1 \implies A_2\mathbf{x} + \mathbf{b}_2 + \delta_2 \end{cases}$$

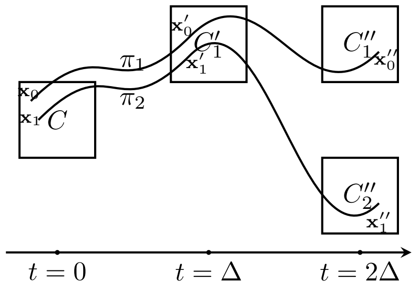


Fig. 6: The $k = 2$ refinement further splits the data set $\mathcal{D}_1$ into two sets $\mathcal{D}_{11} = \{\pi_1\}$ and $\mathcal{D}_{12} = \{\pi_2\}$ and $R_{(C,C'_1)}$ now has two affine maps, non-deterministically defining the system behavior.

Finally, we generalize the relational models to k-relational PWA models. A k-relational model is constructed by using k length *connected* segmented trajectories. A segmented trajectory $\mathbf{S}_\pi$ is *connected* iff its cost $\text{COST}(\mathbf{S}_\pi) = 0$.

Two $\mathbf{S}_\pi : \langle \pi_{t_1}, \dots, \pi_{t_k} \rangle$ and $\mathbf{S}'_\pi : \langle \pi'_{t_1}, \dots, \pi'_{t_k} \rangle$ are *similar* if their trajectory segments have the same sequence of cell traversals $\langle C_1, C_2, \dots, C_k \rangle$, i.e.

$$\forall i \in \{1 \dots k\}.start(\pi_{t_i}) \in C_i \wedge start(\pi'_{t_i}) \in C_i \wedge end(\pi_{t_i}) \in C_{i+1} \wedge end(\pi$$

For every cell C in the reachability graph, we construct a data set $\mathcal{D}$ by collecting π_{t_i}, such that they are the first segment of k length *connected* and *similar* segmented trajectories.

Fig. 6 illustrates the $k = 2$ refinement on Fig. 5.

V. Bounded Model Checking Black Box Systems

We now describe the entire procedure in three steps.

1) Given a black box system S specified by SIM_S, we use scatter-and-simulate with (Δ-length) trajectory segments to sample a finite reachability graph $\mathcal{H}(\Delta)$.

2) Using regression, we enrich the graph relations and annotate the edges with the estimated affine maps.
3) Interpreting the directed reachability graph as a transition system, we use a bounded model checker to find fixed length violations to safety properties.

VI. An Example: Van der Pol Oscillator

We now describe the above using the Van der Pol oscillator benchmark presented in Sec. **??**. Simulations generated using uniformly random samples are shown in Fig. VI. We want to check the property P_3, indicated by the red box, given the initial set indicated by the green box. The abstraction is defined by $Q_{0.2}(\mathbf{x})$, which results in an evenly gridded state-space, where each cell is of size 0.2×0.2 units. Scatter-and-simulate is then used to construct the abstract graph G, using 2 samples per cell and the time step $\Delta = 0.1s$. The complete process follows.

1) *Abstract* Consider an implicit abstraction induced by the quantization function $Q_\epsilon(\mathbf{x})$. The corresponding reachability graph $\mathcal{H}(\Delta)$ (with time step Δ) is given by $\left\langle \mathcal{C}, \overset{\Delta}{\rightsquigarrow}, \mathcal{C}_0, \mathcal{C}_u \right\rangle$.
2) *Discover*: Using scatter-and-simulate, enumerate a finite number of cells (vertices) and the relations (edges) of the graph $\mathcal{H}(\Delta)$. Associate the set of generated trajectory segments with their respective originating cells using a map $D : C \mapsto \{\pi | start(\pi) \in C\}$. The discovered abstraction is show in Fig. VI. As mentioned, red cells are unsafe cells and green cells are initial cells.
3) *Relationalize* For each cell C, perform regression analysis on the respective set of trajectory segments $D(C)$, and compute a set of affine relations $R_{C,C'}(\mathbf{x}, \mathbf{x}')$ between $\mathbf{x} \in C$ and $\mathbf{x}' \in C'$ s.t., edge$(C, C') \in \mathcal{H}(\Delta)$. Annotate each edge in the graph $\mathcal{H}(\Delta)$ with the respective relation.
Fig. VI shows the cells and the trajectory segments which are part of the data sets constructed using the 1-relational modeling. Against each cell, its unique identifier (integer co-ordinate) is mentioned. Finally, Fig. VI and Table I show the enriched graph G^R with its transition relations. Note that self loops result when an observed trajectory segment has its *start* and *end* states in the same cell.
4) *Model Check* The graph $\mathcal{H}(\Delta)$ can now be viewed as a transition system $\langle C, \mathbf{x}, \mathcal{T}, C_0, \mathcal{X}_0 \rangle$, where $C \in vertices(\mathcal{H}(\Delta))$ and $\mathbf{x} \in \mathcal{X}$. A transition $\tau \in \mathcal{T}$ is of the form $\langle C, C', \rho_\tau \rangle$, where $\rho_\tau(\mathbf{x}, \mathbf{x}') \subseteq \{R_{(C,C')}(\mathbf{x}, \mathbf{x}') | (C, C') \in edge(\mathcal{H}(\Delta))\}$.
5) *Check Counter-example* The infinite state (but finite location) transition system can be model checked to find a concrete counter-example, which if exists, can indicate the existence of a similar trace in the original black-box system S. The latter check is carried out as before, using the numerical simulation function SIM. For the given example, the model checker is unable to find a counter-example.

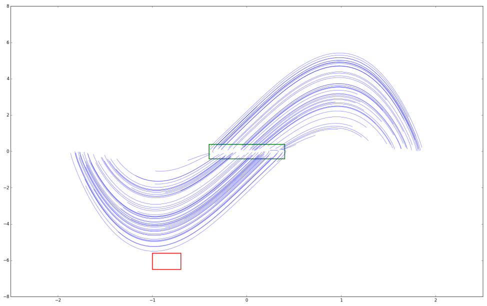


Fig. 7: Van der Pol: continuous trajectories. Red and green boxes indicate unsafe and initial sets.

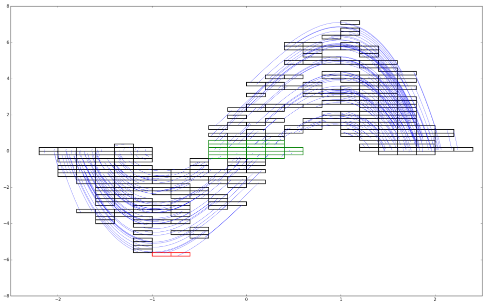


Fig. 8: The discovered abstraction $\mathcal{H}(0.1)$. Red cells are unsafe cells and green cells are initial cells.

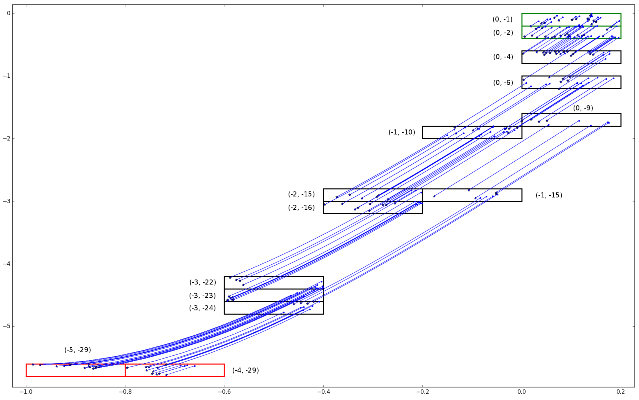


Fig. 9: Cells and trajectory segments used by 1-relational modeling.

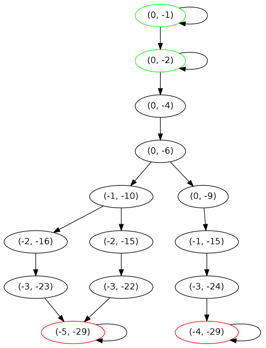


Fig. 10: Enriched graph G^R. The affine maps f for the transition relations are show in Table I.

TABLE I: PWA model computed using OLS. The affine model for each edge (C, C') in the graph Fig. VI is given by $x' \in Ax + b + \delta$, where δ is a vector of intervals.

C	C'	A	b	δ
(0, -1)	(0, -1)	$\begin{bmatrix} 0.99 & 0.12 \\ -0.12 & 1.63 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(0, -1)	(0, -2)	$\begin{bmatrix} 0.99 & 0.12 \\ -0.10 & 1.63 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(0, -2)	(0, -2)	$\begin{bmatrix} 0.99 & 0.12 \\ -0.09 & 1.63 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(0, -2)	(0, -4)	$\begin{bmatrix} 0.99 & 0.12 \\ -0.07 & 1.62 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(0, -9)	(-1, -15)	$\begin{bmatrix} 0.99 & 0.12 \\ -0.09 & 1.61 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.04 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0.01] \end{bmatrix}$
(-1, -15)	(-3, -24)	$\begin{bmatrix} 0.95 & 0.12 \\ -1.29 & 1.47 \end{bmatrix}$	$\begin{bmatrix} -0.01 \\ -0.41 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0.01] \end{bmatrix}$
(-3, -24)	(-4, -29)	$\begin{bmatrix} 1.11 & -0.17 \\ -1.71 & 0.66 \end{bmatrix}$	$\begin{bmatrix} -1.07 \\ -3.31 \end{bmatrix}$	$\begin{bmatrix} [-0.04, 0.04] \\ [-0.07, 0.09] \end{bmatrix}$
(-4, -29)	(-4, -29)	$\begin{bmatrix} 0.99 & 0.01 \\ -0.41 & 1.02 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.28 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(-1, -10)	(-2, -16)	$\begin{bmatrix} 0.97 & 0.12 \\ -0.71 & 1.56 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.11 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0.01] \end{bmatrix}$
(-1, -10)	(-2, -15)	$\begin{bmatrix} 0.97 & 0.12 \\ -0.70 & 1.58 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.08 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0.01] \end{bmatrix}$
(-2, -16)	(-3, -23)	$\begin{bmatrix} 0.92 & 0.12 \\ -1.84 & 1.39 \end{bmatrix}$	$\begin{bmatrix} -0.02 \\ -0.73 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(-3, -23)	(-5, -29)	$\begin{bmatrix} 3.14 & -0.81 \\ -0.96 & 0.23 \end{bmatrix}$	$\begin{bmatrix} -3.13 \\ -4.99 \end{bmatrix}$	$\begin{bmatrix} [-0.04, 0.03] \\ [-0.03, 0.04] \end{bmatrix}$
(-2, -15)	(-3, -22)	$\begin{bmatrix} 0.93 & 0.12 \\ -1.75 & 1.40 \end{bmatrix}$	$\begin{bmatrix} -0.02 \\ -0.68 \end{bmatrix}$	$\begin{bmatrix} [0, 0] \\ [0, 0] \end{bmatrix}$
(-3, -22)	(-5, -29)	$\begin{bmatrix} 3.08 & -0.72 \\ -1.69 & 0.40 \end{bmatrix}$	$\begin{bmatrix} -2.77 \\ -4.57 \end{bmatrix}$	$\begin{bmatrix} [-0.02, 0.02] \\ [-0.01, 0.02] \end{bmatrix}$

A. Search Parameters

The search parameters for S3CAM-R include the parameters of S3CAM: N, ϵ and Δ. Additionally, they also include the maximum error budget δ_{max} for OLS and the maximum length of segmented trajectory for building k-relational models.

We have already discussed the effects of N, ϵ and Δ on S3CAM's performance. However, they also have an effect on relational modeling. A finer grid with small cells produces more accurate models than a coarser grid with large cells. Similarly, small time length trajectory segments result in more accurate modeling.

Maximum Model Error ($\boldsymbol{\delta_{\mathbf{max}}}$) . Given a δ_{max}, we keep increasing k during the k-relational modeling process till $\delta_i \leq \delta_{max}$ is satisfied. A k_{max} is introduced to bound the longest segmented trajectory which can be considered.

B. Reasons for Failure

The approach can fail to find a counter-example, even when it exists, in one of three ways.

- *S3CAM Fails* No abstract counter-example is found by S3CAM. We can remedy it by increasing search budgets and/or restarting.
- *BMC Fails* An abstract counter-example is found, but the BMC fails to find a concrete counter-example in the PWA relational model. The failure can be attributed to either (a) a spurious abstract counter-examples or (b) a poorly estimated model. The former can be addressed by restarting but the latter requires that the maximum model error δ_{max} be decreased.
- *Inaccurate PWA Modeling* An abstract counter-example is found, and it is successfully concretized in the PWA relational model by the BMC. However, it is not reproducible in the black box system. This happens when the PWA relational model is not precise enough.

VII. Implementation and Evaluation

The implementation was prototyped as S3CAM-R, an extension to our previously mentioned tool S3CAM (Chapter **??**). OLS regression routines were used from Scikit-learn [7], a Python module for machine learning. SAL [8] with Yices2 [3] was the model checker and the SMT solver.

We used S3CAM to discover the abstraction graph G, which was then trimmed of the nodes which did not contribute to the error search (nodes from which error nodes were not reachable). In our experiments we used 1-relational modeling to create the transition system from G. Using SAL, we checked for the given safety property. If a concrete trace was obtained from the BMC, it was further checked for validity by simulating using SIM to see if it was indeed an error trace. If not, 100 samples from its associated abstract state was sampled to check the presence of a counter-example in it neighborhood. This can be further extended by obtaining different discrete sequences or discrete trace sequences from the model checker.

We tabulate our preliminary evaluation in Table II. We used few of the benchmarks described in Chapter **??**, including the Van der Pol oscillator, Brusselator, Lorenz attractor and the Navigation benchmark. As before, we ran S3CAM-R 10 times with different seeds and averaged the results. We tabulate both the total time taken and the time taken by SAL to compute the counter-example and compare against a newer implementation of S3CAM. Note that different abstraction parameters are used for S3CAM and S3CAM-R, due to the differences in which they operate.

The results show promise, but clearly S3CAM performs better. Also, S3CAM-R timed out on some benchmarks like Nav $\mathcal{P}_R$, Nav $\mathcal{P}_S$ and the bouncing ball. However, we need to explore more benchmarks to be conclusive. S3CAM-R's performance can be explained by the difficulty in finding a good abstraction (Q_ϵ), which needs to be fine enough, to obtain good prediction models but coarse enough, to be manageable by the current SMT solvers. Specifically, to obtain a good quality counter-example from the model checker (and avoid false positives), the transition system should be created from models with high accuracy, for which a finer abstraction is required. However, the exploration of a finer abstraction is exponentially more complex and results in a much bigger transition system, which the current state of the art bounded model checkers (which use SMT solvers) can not handle under reasonable resources.

VIII. Conclusion and Future Work

We have presented another methodology to find falsifications in black box dynamical systems. Combining the ideas from abstraction based search (Chapter **??**), with our previous relational abstractions [10], we proposed k-relational modeling to compute richer abstractions. Simple linear regression can be used to estimate the local dynamics of the trajectory segments and compute PWA relational models which can be interpreted as a transition system. These can then be checked for safety violations using an off-the-shelf bounded model checker. Finally, we implemented the ideas as a tool S3CAM-R and demonstrated its performance on a few examples.

A. Improvements

The approach seems promising as it avoids an expensive refinement step. But, before it can be applied to more complex systems, we need to overcome its shortcomings. As a future extension, we are investigating ideas to improve the performance of the underlying three main sub-processes, namely, abstraction, modeling, and BMC. We detail these below.

1) *Abstraction* The abstract domain we have used, has been confined to the interval or rectangular domain. Such a domain is very coarse when compared to more general polyhedral domains, using which, we can gain better precision. Due to polyhedral domains being computational expensive, their integration in S3CAM needs to be explored thoroughly.
2) *Modeling* We have used k-relational modeling to increase the precision of the enriched abstractions, however, there are several other refinement strategies that remain to be explored. This includes refining the abstract states by using guard predicates (similar to predicate abstraction), using non-linear templates in parametric regression to compute more precise quadratic relations, and using an adaptive time step instead of a fixed one to address the non-linearities due to a long time horizon.
3) *Model Checking* As of now, we use SMT solvers to search for concrete counter-examples. Although they are very efficient, owing to extensive engineering effort, reachability of transition systems resulting from dynamical systems remains a difficult problem. As the time horizon of the safety property increases, the possible combinations of discrete transitions increase exponentially. Hence, to find a counterexample which is a sequence of discrete transitions over a 'long' time horizon is not tractable for most, but the simplest of dynamical systems. Instead, we can use linear programming solvers, by enumerating each path in the graph G^R. Note that the constraints are all linear (conjunctions) along an abstract path. However, in the worst case, the number of paths in a graph can be of the order $n!$ where n is the number of vertices of a graph. Hence, this will not be feasible, unless, we can prioritize paths by using a triage process similar to one used by S3CAM and using a budget on the maximum number of paths.

   Another approach to address the issue would be to use an adaptive time discretization technique, where relations over both shorter and longer time steps are computed, and the SMT solver can 'select' the time step precise enough to find a counter-example.
4) *SMT Solvers* One major impediment to our approach is the fact that SMT solvers use the theory of reals with *exact precision*. This is important for verification approaches, but can be relaxed for the problem of falsification. An SMT solver which uses approximate reasoning but returns robust counter-examples will be as useful, and perhaps more efficient.

B. Data Driven Analysis

Finally, we would like to mention that our approach is 'simulation' driven. It can be easily modified to be 'data driven', by working with a fixed set of data. Given a data set, we then need to automatically find a good abstraction and a high fidelity PWA transition system using which, we can summarize the black box hybrid dynamical system. Apart

TABLE II: Avg. timings for benchmarks. The **BMC** column lists time taken by the BMC engine. The total time in seconds (rounded off to an integer) is noted under **S3CAM-R** and **S3CAM**. TO signifies time $> 5hr$, after which the search was killed.

Benchmark	BMC	S3CAM-R	S3CAM
Van der Pol ($\mathcal{P}_3$)	<1	130	24
Brusselator	<1	4	2
Lorenz	<1	122	35
Nav ($\mathcal{P}_P$)	4480	9423	603
Nav ($\mathcal{P}_Q$)	970	8105	546
Nav ($\mathcal{P}_R$)	-	TO	2003
Nav ($\mathcal{P}_S$)	-	TO	2100
Bouncing Ball	-	TO	450

from model checking the transition system, one can extend it to the analysis of SDCS. More specifically, by combining the transition system model of a plant with the control software, a model checker like CBMC [**?**] can be used to do a closed loop symbolic analysis of the SDCS. This can be an alternative to S3CAM-X.

References

[1] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification*, pages 258–263. Springer, 2013.

[2] M. Colón, S. Sankaranarayanan, and H. Sipma. Linear invariant generation using non-linear constraint solving. In *CAV*, volume 2725 of *LNCS*, pages 420–433. Springer, July 2003.

[3] B. Dutertre. Yices 2.2. In *International Conference on Computer Aided Verification*, pages 737–744. Springer, 2014.

[4] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[5] S. Gulwani and A. Tiwari. Constraint-based approach for hybrid systems. In *CAV*, volume 5123 of *LNCS*, pages 190–203, 2008.

[6] S. Mover, A. Cimatti, A. Tiwari, and S. Tonetta. Time-aware relational abstractions for hybrid systems. In *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on*, pages 1–10. IEEE, 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[8] J. Rushby, P. Lincoln, S. Owre, N. Shankar, and A. Tiwari. Symbolic analysis laboratory (sal). Cf. http://www.csl.sri.com/projects/sal/.

[9] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *CAV*, volume 6806 of *LNCS*, pages 686–702. Springer, 2011.

[10] A. Zutshi, S. Sankaranarayanan, and A. Tiwari. Timed relational abstractions for sampled data control systems. In *Computer Aided Verification*, pages 343–361. Springer, 2012.

Acknowledgment

The preferred spelling of the word acknowledgment in America is without an e after the g. Avoid the stilted expression, One of us (R. B. G.) thanks . . . Instead, try R. B. G. thanks. Put sponsor acknowledgments in the unnumbered footnote on the first page.