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

ABSTRACT

Models of embedded control systems are often so complex that they can be treated as effectively black-box systems. Sampling-based methods for testing such systems use numerical simulations to search for violations of properties (specified in a suitable requirement formalism, such as temporal logic). This constrains them to a 'best effort' category unlike rigorous verification techniques, which however require a model amenable to symbolic analysis. This work tries to bridge the gap by presenting an automatic falsification technique which learns a data-driven abstraction of the black box system before searching for a violation. It proceeds by (a) modeling the behavior of the system using a piecewise affine (PWA) discrete-time *relational* model and, (b) encoding the search for a violation in the abstract model as a bounded model checking problem. The bounded model checking query in this instance can be viewed as a disjunction of conjunctions of linear constraints (i.e. linear programs) and can be solved directly using off-the-shelf SMT solvers or by multiple runs of a linear programming solver. For any counter-example found in the abstraction, we check the existence of a corresponding violation in the original system. We demonstrate the efficacy of our technique on a few dynamical systems examples.

1. 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 [34], 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. Due to modeling errors, this might not be reproducible in the original black-box system. 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.

2. RELATED WORK

Sampling based falsification techniques for hybrid systems improve upon the commonly used testing approaches. However, a wide gap between falsification and verification approaches. The current state-of-the-art approaches (Cf. related work [21]) for testing hybrid systems are based on numerical simulations and are guided by robustness: S-Taliro [2] and Breach [10], and RRT (Rapidly-exploring Random Tree): [9, 20] or both [11]. An alternative approach based on multiple shooting and CEGAR (Counterexample Guided Abstraction Refinement), was proposed in [34]. Unlike the previous approaches which directly use numerical simulations, it searches an implicit abstraction by using the computed trajectories to enumerate the relations between abstract states. The resulting graph over relations is searched for paths corresponding to abstract counter-examples. In this paper, we show how this search procedure can be combined with the identification of the affine dynamics associated with each relation of the abstraction, effectively computing a 'relational abstraction'. We now overview the several existing PWA

identification techniques for hybrid systems.

Identification of hybrid systems. Discrete time - continuous state piece-wise affine models are often used to model both continuous and hybrid dynamical systems. They are very expressive and can model non-linear continuous dynamics (with arbitrary accuracy) [31] and a large family of hybrid systems [15]. In the past, several approaches have been put forth for the identification of PWA models [22]. They are based on Bayesian methods [16], bounded-error methods [1,5,6,25], clustering based methods [13] and algebraic methods [30]. We propose a simpler approach based on fixed hyper-rectangular domains, also explored in [7], but parameterized by the size of the rectangles ϵ and the time discretization step Δ. [[other tech.]]

Formal analysis of pwa systems. Formal verification of pwa systems using reachability analysis has been proposed in [3,17,33], and extended to model checking temporal properties in [4,32]. PWA systems can be equivalently translated to non-deterministic infinite state transition systems and then model checked. Related approaches using a bisimilar quotient have been proposed in [23, 29, 32]. Finally, because hybrid automata models can equivalently represent pwa systems, existing falsification and verification tools can be used for their analysis.

Relational abstractions were first introduced by [27] for abstracting away the continuous dynamics in hybrid automata by discrete relations, resulting in an infinite state transition system. Both time independent [27] and time dependent relations [19, 35] 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. Properties of relational abstractions can be verified using k-induction or falsified using bounded model checkers.

3. PRELIMS

In this section, we present the model of the hybrid dynamical system under test, and the background for pwa relational abstractions.

The System under test S is assumed to be a hybrid dynamical system. Its state is denoted by $\mathcal{X} \subseteq \mathcal{Q} \times \mathbb{R}^n$, where $\mathcal{Q}$ is a finite set of discrete modes. Let $\mathcal{U} : [0, T] \to \mathbb{R}^k$ be the set of input signals to S over a finite time horizon T.

For the purpose of falsification, S is viewed as a black-box equipped with a forward simulation map $\text{SIM}^S_\Delta : (\mathbf{x}, \mathbf{u}, t) \mapsto \mathbf{x}$. A simulation step involves computing the unique reachable state $\mathbf{x} \in \mathcal{X}$ in time t from a given state input pair $(\mathbf{x}, \mathbf{u})$. We formalize this using the notion of time parameterized reachability relations $\xrightarrow{t,\mathbf{u}}_S \subseteq \mathcal{X} \times \mathcal{X}$ detailed in [34].

DEFINITION 3.1 (SYSTEM EVOLUTION). *The system under test S evolves in the set of infinite hybrid state-space $\mathcal{X}$ under the effect of inputs $\mathbf{u} \in \mathcal{U}$. Its evolution is defined by the simulation function $\text{SIM}^S_\Delta : \mathcal{X} \times \mathcal{U} \times \mathbb{R}^+ \mapsto \mathcal{X}$, where $\text{SIM}^S_\Delta(\mathbf{x}, \mathbf{u}, \Delta_t)$ maps a current state $\mathbf{x}$ at time t to a new state $\mathbf{x}'$ at time $t + \Delta_t$, under a constant input $\mathbf{u}$. Finally, $\mathbf{x}' = \text{SIM}^S_\Delta(\mathbf{x}, u, t)$ iff $\mathbf{x} \xrightarrow{t,u} \mathbf{x}' \in \xrightarrow{t,\mathbf{u}}_S$.*

System Assumptions: We require that S can always be simulated forward in time with deterministic results. This in turn requires existence and uniqueness of trajectories over a finite time horizon $[0, T]$, which can be guaranteed by Lipschitz continuity of the vector field in each underlying hybrid mode and ruling away issues such as finite escape times [18]. Furthermore, as the relations abstract the solution of a continuous time, foward deterministic hybrid dynamical system, they are deterministic, causal and satisfy the semi-group property. These assumptions are justified as for the dynamical systems we address in this presentation. Finally, we assume full observability of the system state, validate our results only against the SIM_Δ function (as the analytic closed-form representation of the system dynamics is assumed to be unavailable). From here on, we drop the subscript S.

3.1 Learning Dynamics using Simple Linear Regression

Best effor falsification works by enumerating the reachability relations $\to$ within a given budget. Depending upon the specific instance of the problem, the budget can be prohibhitive, requiring exponential computational and storage resources. Instead, we proceed by summarizing these relations by their corresponding affine maps using linear regression. The learnt analytical forms are represented as $\mathbf{x}' = A\mathbf{x} + B\mathbf{u} + \delta$, where A and B are matrices and δ the estimated error.

In statistics, linear regression is the problem of finding an affine function or *predictor*, which can 'best' summarize the relationship between the given set of observed input $\mathbf{x}$ and output $\mathbf{y}$ vectors. The notion of 'best' is formally captured using a loss function which can be non-linear. We use the commonly used loss function *ordinary least squares* (OLS) for this presentation.

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 choice 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.

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

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

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

The solution of OLS can be analytically computed as

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

where X is the matrix representing the horizontal stacking of all $\hat{\mathbf{x}}$. The details can be found in standard texts on learning and statistics [14].

Given a time invariant dynamical system $\mathbf{x}' = \text{SIM}_\Delta(\mathbf{x}, \Delta)$, 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 $\{(\mathbf{x}, \mathbf{x}')|\mathbf{x}' = \text{SIM}_\Delta(\mathbf{x}, \Delta)\}$ 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 SIM_Δ. This results in a piece wise approximation, as described in the next section.

3.2 Piecewise Affine (PWA) Transition System

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

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

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

We can now use a PWA transition system ρ to approximate the behavior of a hybrid dynamical system S defined by a simulator SIM^S_Δ [1]. 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 hyper-planes 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 relation f_τ can be represented using matrix A and an offset vector $\mathbf{b}$ as $f_\tau \subseteq \{(\mathbf{x}, \mathbf{x}')|\mathbf{x}' = A\mathbf{x} + \mathbf{b}\}$. Finaly, we incorporte an approximate non-deterministic error term δ, defined as a vector of intervals $[\delta_{\min}, \delta_{\max}]$.

$$f_\tau \subseteq \{(\mathbf{x}, \mathbf{x}')|\mathbf{x}' = A\mathbf{x} + \mathbf{b}\}.$$

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

DEFINITION 3.2 (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*

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

where g_i and g'_i are affine predicates and f_i are affine relations 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 $\mathbf{x}' = \rho(\mathbf{x})$.

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 modeling errors, and from here on, we do not consider such cases.

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

3.3 Bounded Model Checking (BMC)

The PWA transition system is a finite location, infinite state transition system. As all the guards and transitions are affine, a bounded reachability query is equivalent to checking all combinaions of discrete locations, where each combination can be summarized as a linear program. As the time is bounded, the number of combinations are finite, but, exponential in number.

It should be noted that an explicit execution of the transition system represents a directed acyclic graph, with each location being a node, and a directed edge from each node to all nodes reachable in forward time. A path on this graph is then a linear program, with each branching node corresponsing to a logical disjunction.

Temporal properties of pwa transitions systems can be checked using off-the-shelf model checkers like SAL [26], which can reason over infinite state transitions systems using SMT solvers. Furthermore, lazy SMT solvers can be employed for this specific problem instance to achieve better efficiency [28]. We now show how relational pwa abstractions can be viewed as pwa transition systems, and how the problem of falsification can be answered by a BMC query.

4. PWA RELATIONAL MODELING

4.1 PWA Relational Abstraction

Given an abstraction, we can define its enrichment

We now briefly overview the approach, which consists of three steps

Discover the abstraction enriches the abstraction formulates a BMC query concretizes

abstraction defined by a quantization function Q_ϵ time step Δ

Given a numerical simulator SIM_Δ 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 [34]) 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 (used in [34]), 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.

4.2 Graph Abstraction

DEFINITION 3.4 (ABSTRACT STATE GRAPH). Let âĹĘ > 0 be a fixed time step. The abstract state graph H(âĹĘ) for time step âĹĘ is defined by the set of vertices C, and edges (Ci, Cj) whenever Ci âĹĘ Cj . Let C0 denote the collection of initial cells in C, i.e., cells Ci such that Ci âĹl' X0 6= âĹĔ. Further, let Cu denote the set of unsafe cells, or cells Cj such that there is a state xj âĹĹ Cj that reaches an unsafe state y âĹĹ Xf within time 0 âL'd' tj < âĹĘ. The abstraction H(âĹĘ) is given by D C, âĹĘ , C0, Cu E .

4.3 Enriched Graph Abstraction

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.

The enriched graph G^R captures the underlying local forward dy-

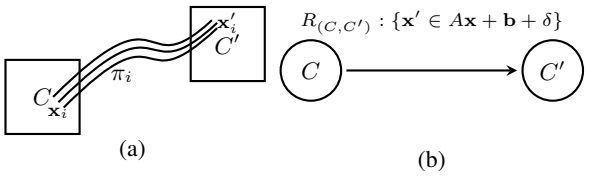

Figure 1: (a) Trajectory segments π_i are used to compute the relation $R_{(C,C')}$ that annotates the edge in (b). $R_{(C,C')} : \{\mathbf{x}' \in A\mathbf{x}+\mathbf{b}+\delta\}$ is an interval affine relation defined by an affine map (matrix A and vector $\mathbf{b}$) and an error interval (vector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

namics 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_Δ, 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* [8] 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.

5. RELATIONAL MODELING

We now describe how relational models can be computed for a given black box system by enriching the abstract reachability graph. 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.

5.1 Abstract Enriched Graph

The existential abstraction relation in [34] 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. **??**).

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

Figure 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.

5.2 k - Relational Modeling

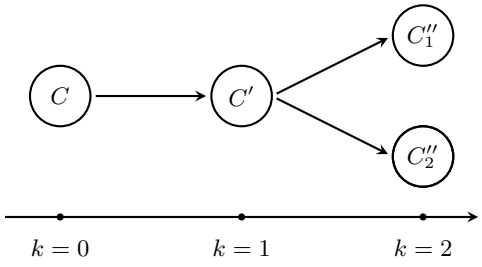

Figure 3: Nodes/Cells of G shown along with increasing values of k.

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) $\mathbf{x}(t + \Delta) \in C' \wedge \mathbf{x}(t + 2\Delta) \in C''_1$

(b) $\mathbf{x}(t+\Delta) \in C' \wedge \mathbf{x}(t+2\Delta) \in C''_2$

- $k = n$: 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

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

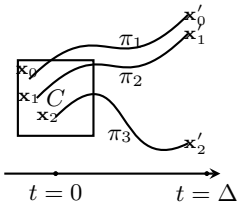

Figure 4: All the trajectory segments will be used to construct the model; $\mathcal{D} = \{\pi_1, \pi_2, \pi_3\}$.

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 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.

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

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

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.

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

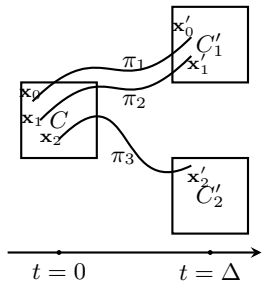

Figure 5: The data gets split into two sets $\mathcal{D}_1 = \{\pi_1, \pi_2\}$ and $\mathcal{D}_2 = \{\pi_3\}$ and two relations: $R_{(C,C'_1)}$ and $R_{(C,C'_2)}$, each with one affine map, are constructed.

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

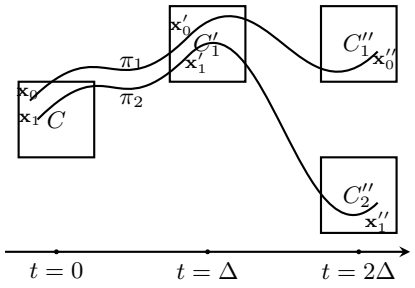

Figure 6: The $k = 2$ refinement further splits the data set $\mathcal{D}_1$ into two sets $\mathcal{D}_{11} = \{\pi_1\}$ and $\mathcal{D}_{12} = \{\pi_2\}$ and $R_{(C,C'_1)}$ now has two affine maps, non-deterministically defining the system behavior.

k-Relational Model

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

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

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.

5.3 Learning Affine Function from Trajectory Segments

The SIM_Δ function can be used to generate a data set of relations parameterized by time Δ_t, for different values of states $\mathbf{x}_i$ and inputs $\mathbf{u}_i$ as $\mathcal{D}_{\Delta_t} = \{(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{u}_i) | \mathbf{x}'_i = \text{SIM}_\Delta(\mathbf{x}_i, \mathbf{u}_i)\}$. Moreover, by repeatedly applying SIM_Δ, a discrete time series of a finite length N can be generated from a given state $\mathbf{x}_1$ and a sequence of inputs $\langle \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n \rangle$ as shown below.

$$\langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle \text{ where } \mathbf{x}_{i>1} = \text{SIM}_\Delta(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}, \Delta_t)$$

We now introduce a refinement of the data set $\mathcal{D} \subseteq \mathcal{D}$ by incrementally observing the future states in the time series.

$$\mathcal{D}_{\Delta_t} = \{(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{u}_i) | \mathbf{x}'_i = \text{SIM}_\Delta(\mathbf{x}_i, \mathbf{u}_i)\}$$

$$\mathcal{D}^{C_i} \subseteq \mathcal{D}_t$$

$$\mathcal{D}^{C_i}$$

k-Refinement of the data set $\mathcal{D}^k$. $\mathcal{D}_k = \{(\mathbf{x}, \mathbf{x}', \mathbf{u}) | \mathbf{x}' = \text{SIM}_\Delta(\mathbf{x}, \mathbf{u}, k\Delta_t)\}$

state-input pairs of the form $(\mathbf{x}, \mathbf{x}')$ data set equivalence traj segment D_c,

6. BOUNDED MODEL CHECKING FOR BLACK BOX SYSTEMS

We now describe the entire procedure in three steps.

1. Given a black box system S specified by SIM^S_Δ, 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.

7. AN EXAMPLE: VAN DER POL OSCILLATOR

We now describe the above using the Van der Pol oscillator benchmark presented in [34]. Simulations generated using uniformly random samples are shown in Fig. 7. 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. 7. As mentioned, red cells are unsafe 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. 7 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. 7 and Table 1 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_Δ. For the given example, the model checker is unable to find a counter-example.

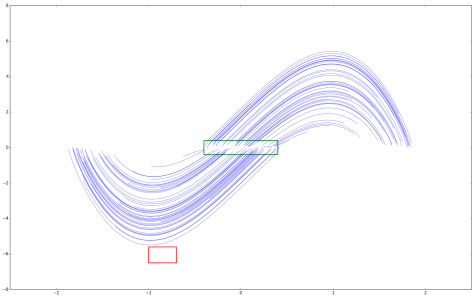

Figure 7: Van der Pol: continuous trajectories. Red and green boxes indicate unsafe and initial sets.

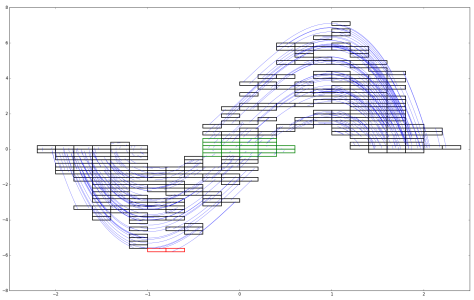

Figure 8: The discovered abstraction $\mathcal{H}(0.1)$. Red cells are unsafe cells and green cells are initial cells.

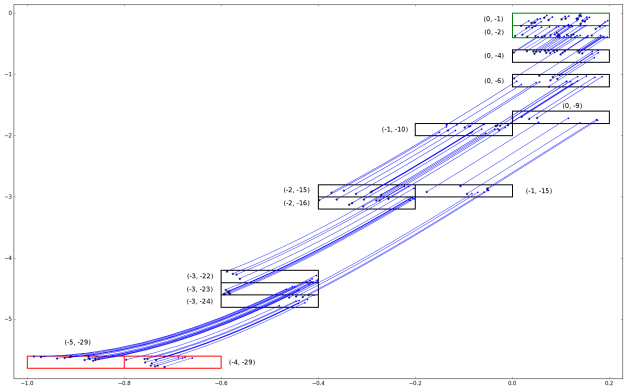

Figure 9: Cells and trajectory segments used by 1-relational modeling.

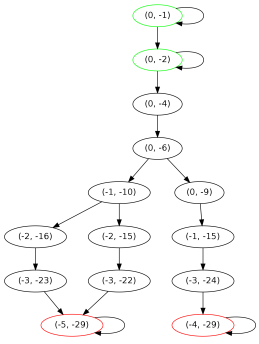

Figure 10: Enriched graph G^R. The affine maps f for the transition relations are show in Table 1.

Table 1: PWA model computed using OLS. The affine model for each edge (C, C') in the graph Fig. 7 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)	$\begin{bmatrix} 3.08 & -0.72 \\ -1.69 & 0.40 \end{bmatrix}$	$\begin{bmatrix} -2.77 \\ -4.57 \end{bmatrix}$	$\begin{bmatrix} [-0.02,0.02] \\ [-0.01,0.02] \end{bmatrix}$

7.1 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 ($\delta_{\max}$) . 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.

7.2 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.

8. IMPLEMENTATION

The implementation was prototyped as S3CAM-R, an extension to our previously mentioned tool S3CAM [34]. OLS regression routines were used from Scikit-learn [24], a Python module for machine learning. SAL [26] with Yices2 [12] 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_Δ 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 2. We used few of the benchmarks described in [34], 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.

Table 2: 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	PWA Rel + BMC	PWA Rel + LP	S3CAM
Van der Pol ($\mathcal{P}_3$)	130	0	24
Brusselator	4	0	2
Lorenz	122	0	35
Nav ($\mathcal{P}_P$)	TO	2100	603
Nav ($\mathcal{P}_Q$)	TO	$25m$	546
Nav ($\mathcal{P}_R$)	TO	0	2003
Nav ($\mathcal{P}_S$)	TO	0	2100
Bouncing Ball	TO	0	450

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.

9. CONCLUSIONS

We have presented another methodology to find falsifications in black box dynamical systems. Combining the ideas from abstraction based search [34], with our previous relational abstractions [35], 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.

[[extension to LTL/MTL props]]

10. REFERENCES

[1] R. Alur and N. Singhania. Precise piecewise affine models from input-output data. In *Embedded Software (EMSOFT), 2014 International Conference on*, pages 1–10. IEEE, 2014.

[2] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. *S-taliro: A tool for temporal logic falsification for hybrid systems*. Springer, 2011.

[3] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[4] G. Batt, C. Belta, and R. Weiss. Model checking genetic regulatory networks with parameter uncertainty. In *International Workshop on Hybrid Systems: Computation and Control*, pages 61–75. Springer, 2007.

[5] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A greedy approach to identification of piecewise affine models. In

International Workshop on Hybrid Systems: Computation and Control, pages 97–112. Springer, 2003.

[6] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005.

[7] S. Billings and W. Voon. Piecewise linear identification of non-linear systems. *International journal of control*, 46(1):215–235, 1987.

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

[9] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[10] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, pages 167–170. Springer, 2010.

[11] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods*, pages 127–142. Springer, 2015.

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

[13] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

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

[15] W. P. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[16] A. L. Juloski, S. Weiland, and W. Heemels. A bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005.

[17] X. D. Koutsoukos and P. J. Antsaklis. Safety and reachability of piecewise linear hybrid dynamical systems based on discrete abstractions. *Discrete Event Dynamic Systems*, 13(3):203–243, 2003.

[18] J. D. Meiss. *Differential Dynamical Systems*. SIAM publishers, 2007.

[19] 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.

[20] T. Nahhal and T. Dang. Test coverage for continuous and hybrid systems. In *Computer Aided Verification*, page 449âĂŞ462, 2007.

[21] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 211–220. ACM, 2010.

[22] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems a tutorial. *European journal of control*, 13(2-3):242–260, 2007.

[23] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[24] 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.

[25] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

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

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

[28] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada. Smc: Satisfiability modulo convex optimization. *ArXiv e-prints*, 2017.

[29] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[30] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 167–172. IEEE, 2003.

[31] C. Wen and X. Ma. A basis-function canonical piecewise-linear approximation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(5):1328–1334, 2008.

[32] B. Yordanov, G. Batt, and C. Belta. Model checking discrete-time piecewise affine systems: application to gene networks. In *Control Conference (ECC), 2007 European*, pages 2619–2626. IEEE, 2007.

[33] B. Yordanov and C. Belta. Formal analysis of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 55(12):2834–2840, 2010.

[34] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, page 5. ACM, 2014.

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

[6] out of 1,000 simulations