Symbolic-Numeric Reachability Analysis of Closed-Loop Control Software

Aditya Zutshi
University of Colorado, Boulder
aditya.zutshi@colorado.edu

Sergio Mover
University of Colorado, Boulder
sergio.mover@colorado.edu

Jyotirmoy V. Deshmukh
Toyota Technical Center, LA
jyotirmoy.deshmukh@toyota.com

Sriram Sankaranarayanan
University of Colorado, Boulder
srirams@colorado.edu

ABSTRACT

In this work we address the problem of findfing behaviors violating a given time bounded LTL property in hybrid dynamical systems. We approach the problem in two distinct steps. First, we approximate the behavior of the underlying system using a piecewise affine discrete time model. The model is incomplete in nature and is computed with respect to the given property. We then encode the falsification search as a bounded model checking query and use an SMT solver to find a counter example. If found, we check the violation to ensure reproducibility in the given hybrid dynamical system.

1. INTRODUCTION

In Chapter **??**, we analyzed systems by restricting ourselves to black box semantics. This enabled us to introduce an approach independent of the system's structure (in practice). By observing the dynamics only locally and sparsely (due to a coarse abstraction), we were able to efficiently find abstract counter-examples. However, to concretize them, we were forced to 'take a closer look' or refine the abstraction. As noted previously, splitting the state-space evenly is an expensive operation in higher dimensions and we would like to avoid it. We address this by using learning techniques to model the given black box system by a PWA relational model.

In this chapter, we propose an approach which uses regression to quantitatively estimate the discovered relations (witnessed by trajectory segments) by affine maps. These maps are summaries of locally observed behaviors, which we incorporate into the sampled reachability graph as edge annotations. The resulting graph can be interpreted as a discrete transition system and model checked for time bounded safety properties. If a violation is discovered, it is expected to be 'close' to a true violation of the system.

2. OVERVIEW

In this chapter, we further our exploration of trajectory segment based methods; and propose another approach for searching safety violations in dynamical systems. Unlike the CEGAR based approach in Chapter **??** which directly uses the sampled reachability graph, we first model quantitatively the local behaviors (represented by edges). Recall that the edges of the graph represent observed trajectory segments, between the respective cells. From this, we can only conclude that a cell is reachable but cannot comment on the underlying dynamics. If however, we estimate the dynamics of the witnessed trajectory segments between two cells, we can model the local dynamics. As we use affine templates to achieve this, it can be compared to trajectory *linearization*.

The dynamics can be either estimated soundly or only approximated with error estimates. For the former, we require the white box model of a system. Using a reach-set estimation tool like FLOW* [1], the reachable states of an abstract state can be soundly computed. As this is not usually the case, in the rest of the chapter we focus on the case of black box systems. Recall that such systems are equipped with a simulation function SIM, but the internal details of the model are opaque to us. We use simple regression to fit an affine map to the locally observed behaviors and use cross-validation to estimate the errors. This results in an interval affine map, approximating a reachable hyper-rectangle in the state-space.

Observe that the directed reachability graph can be interpreted as a discrete transitions system. However, an explicit model checker can only find abstract counter-examples in it. The graph does not have enough information to enable the direct search of concrete counter-examples. We remedy this by incorporating the estimated local dynamics by annotating the graph edges with them. The resulting transition system is rich enough that an off-the-shelf bounded model checker can find concrete violations of a safety property, thus doing away with the expensive refinement process. We now discuss the background required to present our ideas.

3. RELATED WORK

4. PRELIMS

4.1 Relational Abstraction

For continuous dynamical systems, it is hard to directly reason about reachability. Discretization is often employed to transform the systems into a discrete transition system. One idea is to abstract continuous relations by *reachability invariants*. The resulting relations can be either time independent or time dependent. Depending on the property at hand, we can select the appropriate one. The resulting abstractions (usually discrete transitions systems) can be analyzed using model checkers. We briefly discuss the two types of relational abstractions.

Time-less Relational Abstraction

Relational abstractions for hybrid systems were proposed in [8]. Proposed for hybrid automaton models, the idea is to summarize the continuous dynamics of each mode using a relation over reachable states. The resulting relation is time 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 e.g., an abstraction which 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 relations summarize the given system as a transition system. These can be verified using k-induction or falsified using bounded model checkers. The relationalization procedure involves finding suitable abstractions, such as affine abstractions, eigen abstractions and box abstractions [8]. For this, differnt techniques like template based invariant generation [2, 5] can be used.

Timed Relational Abstraction

As the above discussed relations are timeless, we cannot check their timing properties. Moreover, they cannot be easily specialized for time-triggered systems. Timed relational abstractions provide a similar solution by computing relations, but, time dependent ones.

For the case of dynamics defined by affine set of ODEs, timed relations are provided by their solutions $\mathbf{x}(t) = e^{tA}\mathbf{x}(0)$. 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 time period, we obtain linear relations in $\mathbf{x}$. We demonstrated the usefulness of timed relational abstractions on linear systems in [9], and now incorporate the idea to discretize black box dynamical systems.

4.2 Simple Linear Regression

Regression

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 overdetermined; there are more equations than unknowns. Hence, a single affine function cannot be found which satisfies the equation $\forall i \in \{1 \dots N\}.y_i = A\mathbf{x}_i + \mathbf{b}$. Instead, we need to find the 'best' choice for A and $\mathbf{b}$. 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 matrix of coefficients A and an offset vector $\mathbf{b}$, such that the least square error of the affine predictor is minimized for the given data set.

$$\min_{A,b} \sum_{i=1}^{N} (\mathbf{y}_i - (A\mathbf{x}_i + \mathbf{b}))^2$$

The solution of OLS can be analytically computed as

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

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

4.3 Piecewise Affine Model (PWA)

A PWA model is a collection of guarded affine maps which compute the next state of the underlying dynamical system. The affine maps discretize the underlying continuous-time dynamics.

Given a state vector $\mathbf{x} \in \mathbb{R}^n$, a guarded affine map $\mathcal{T} : (g, f)$ defines the discretized consecution rule as a pair of an affine guard predicate $g : C\mathbf{x} - \mathbf{d} \leq 0$ and an affine map $f : A\mathbf{x} + \mathbf{b}$, where $A \in \mathbb{R}^{n \times n}$, $C \in \mathbb{R}^{m \times n}$ are matrices and $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{d} \in \mathbb{R}^m$ are vectors. A guarded affine map is satisfied if its guard is satisfied.

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

DEFINITION 4.1 (PWA MODEL). *Given a dynamical system over a continuous state-space $X \in \mathbb{R}^n$, a PWA model is a map $X \mapsto \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$ from the state-space of the dynamical system to a finite set of guarded affine maps $\{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$. It defines the consecution rule by the affine map of the satisfied $\mathcal{T}$.*

$$\rho = \begin{cases} \mathcal{T}_1 : f_1(\mathbf{x}) & \text{if } g_1(\mathbf{x}) \\ \mathcal{T}_2 : f_2(\mathbf{x}) & \text{if } g_2(\mathbf{x}) \\ \dots & \dots \\ \mathcal{T}_n : f_n(\mathbf{x}) & \text{if } g_n(\mathbf{x}) \end{cases} \tag{1}$$

where $f_i = A_i\mathbf{x} + \mathbf{b}_i$, and $g_i(\mathbf{x}) \equiv C_i\mathbf{x} - \mathbf{d}_i \leq 0$. Abusing the notation, we denote the next state computed by the PWA model as $\mathbf{x}' = \rho(\mathbf{x})$. A PWA model is *deterministic*, iff for every state $\mathbf{x} \in X$, a unique guarded affine map is satisfied. A PWA model is *complete*, iff for every state $\mathbf{x} \in X$, there exists at least one satisfied guarded affine map $\mathcal{T}$.

5. PWA RELATIONAL MODELING

We now describe how PWA relational modeling can be used to enrich discrete abstractions of black box systems. We first describe the idea of enriched relations. We then show how the annotated reachability graph can be interpreted as a PWA model or as a PWA relational model. Finally, we show that both are instances of a k-relational translation from a reachability graph, which we use in our approach.

5.1 Enriched Abstraction

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 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 $\mathbf{b}$ lies between 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 use cross-validation to estimate the error δ and generalize f to an interval affine map as before $f : A\mathbf{x} + \mathbf{b} \pm \delta$.

Finally, we get an abstraction with the below relation

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

We now compute a PWA model for a given black box system by estimating the dynamics using OLS. We can estimate the affine relations between every cell, but as discussed in Chapter **??**, it is a futile approach. Instead, we use the same heuristic as before: scatter-and-simulate, to select the cell relations for estimation. We build the reachability graph as before, but in addition, annotate each edge with the values of estimated A, $\mathbf{b}$, and δ for the respective relation. The reachability graph thus computed, is a transition system. Using off-the-shelf bounded model checkers, we reason about the system's safety properties.

5.2 PWA Model (0-relational)

When $k = 0$, the 0-relational model is a PWA system which only defines the evolution of states $\mathbf{x}$, and does not specify the reachable cell. The guards of the guarded affine maps are defined over cells: $g_i(\mathbf{x}) \equiv \mathbf{x} \in C_i$.

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. For n cells, this results in a PWA model (and a transition system), as shown below.

$$\rho = \begin{cases} \mathcal{T}1 = A_1\mathbf{x} + \mathbf{b}_1 + \delta_1 & \mathbf{x} \in C_1 \\ \mathcal{T}2 = A_2\mathbf{x} + \mathbf{b}_2 + \delta_2 & \mathbf{x} \in C_2 \\ \dots & \dots \\ \mathcal{T}n = A_n\mathbf{x} + \mathbf{b}_n + \delta_n & \mathbf{x} \in C_n \end{cases} \quad (2)$$

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.

5.3 PWA Relational Model (1-relational)

To improve the preciseness of learnt dynamics, we include the reachability relation (as discussed before this section) 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.

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

The resulting 1-relational model is at least as precise as the corresponding 0-relational model.

5.4 k-relational PWA Model

Finally, we generalize the PWA 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.

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

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.

6. BOUNDED MODEL CHECKING FOR BLACK BOX SYSTEMS

We now describe the entire procedure in three steps.

1. Given a black box system specified by SIM, 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.

6.1 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. An increase in the number of samples also increases the accuracy.

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} can be introduced as the bound on the longest segmented trajectory to consider.

6.2 Reasons for Failure

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

- No abstract counter-example is found by S3CAM. This happens when S3CAM fails. We remedy it by increasing search budgets and/or restarting.

- 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-example 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.

- An abstract counter-example is found, and so is its concretization in the PWA relational model by the BMC. However, it is not reproducible in the black box system. This again happens when the PWA relational model is not precise enough.

7. IMPLEMENTATION

8. 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 [6], Python module for machine learning. SAL [7] with Yices2 [3] was the model checker.

We tabulate our preliminary evaluation in Table 1. We use the Van der Pol oscillator and the Brusselator, as described in the previous chapter. 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 S3CAM.

The results are favorable and show promise. However, we need to explore more benchmarks to conclude conclusively.

9. EXPERIMENTAL RESULTS

10. CONCLUSIONS

11. SUMMARY

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 [9], we created enriched abstractions. These can be checked for safety violations using existing bounded model checkers. We used learning techniques to estimate the local dynamics

Table 1: Avg. timings for benchmarks. The **BMC** column lists time taken by the BMC engine. The total time is noted under **S3CAM-R** and **S3CAM**.

Benchmark	BMC	S3CAM-R	S3CAM
Van der Pol ($\mathcal{P}3$)	$0.0s$	$10.0s$	$15.0s$
Brusselator	$0.2s$	$4.0s$	$2.5s$

underlying trajectory segments, and approximated a transition system from a black box system. Finally, we showed our approach on a few examples.

As a future extension, we are working on a specialized BMC for the problem at hand. We intend to explore efficient ways for encoding the PWA relational system.

12. 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] 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.

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

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

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