HyLAA: A Tool For Computing Simulation-Equivalent Reachability of Linear Systems

Parasara Sridhar Duggirala[1] Stanley Bak[2]

[1]University of Connecticut [2]Air Force Research Lab

Introduction

Simulations are extensively used for testing and validating Cyber-Physical Systems. While testing procedures uncover bugs, the state space is uncountable and hence testing for all possible behaviors is impossible. Reachable set computation approaches compute a sound overapproximation of all possible behaviors, but might not always provide concrete counterexamples.

Contributions

- Formulate **simulation-equivalent reachability** as all possible set of states reached by a behavior defined by a specific *fixed-step simulation algorithm* and provide a sound and complete algorithm for computing it.
- Present two new improvements: **invariant constraint propagation** and **adaptive aggregation**. Invariant constraint propagation is used to handle the invariants in modes of hybrid system and adaptive aggregation to handle the exponential growth in the number of successors.
- Implement these techniques in the tool **HyLAA** and demonstrate the scalability of these techniques proposed.

Preliminaries

A *linear hybrid automaton* is defined to be a tuple $\langle Loc, X, Flow, Inv, Trans, Guard\rangle$ where:

Loc is a finite set of locations (also called modes).

$X \subseteq \mathbb{R}^n$ is the state space of the behaviors.

$Flow : Loc \rightarrow AffineDeq(X)$ assigns an affine differential equation $\dot{x} = A_l x + B_l$ for location l of the hybrid automaton.

$Inv : Loc \rightarrow 2^{\mathbb{R}^n}$ assigns an invariant set for each location of the hybrid automaton.

$Trans \subseteq Loc \times Loc$ is the set of discrete transitions.

$Guard : Trans \rightarrow 2^{\mathbb{R}^n}$ defines the set of states where a discrete transition is enabled.

For a linear hybrid automaton, the invariants and guards are given as a conjunction of linear constraints.

Superposition Principle

$$\tau(x_0+\Sigma_{i=1}^m \alpha_i v_i, t) = \tau(x_0, t)+\Sigma_{i=1}^m \alpha_i(\tau(x_0+v_i, t)-\tau(x_0, t)).$$

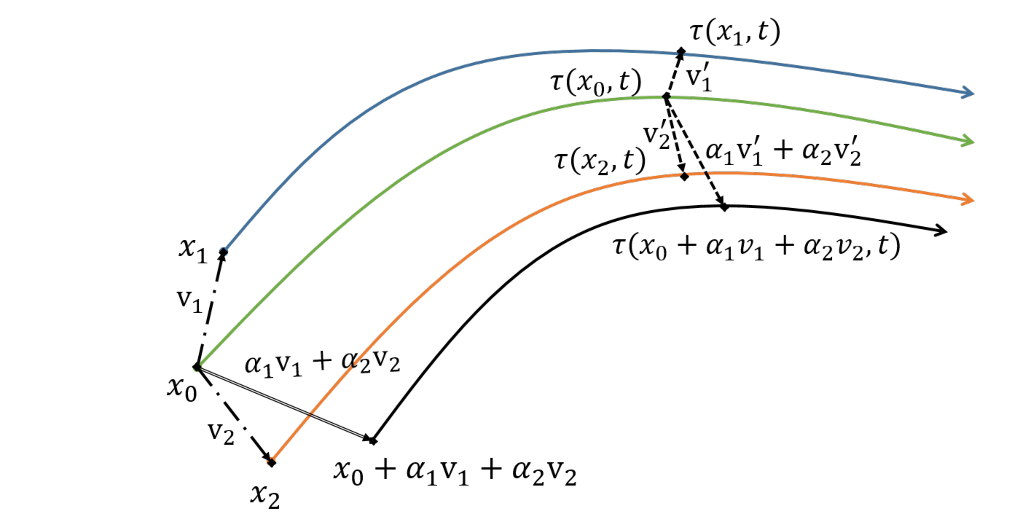
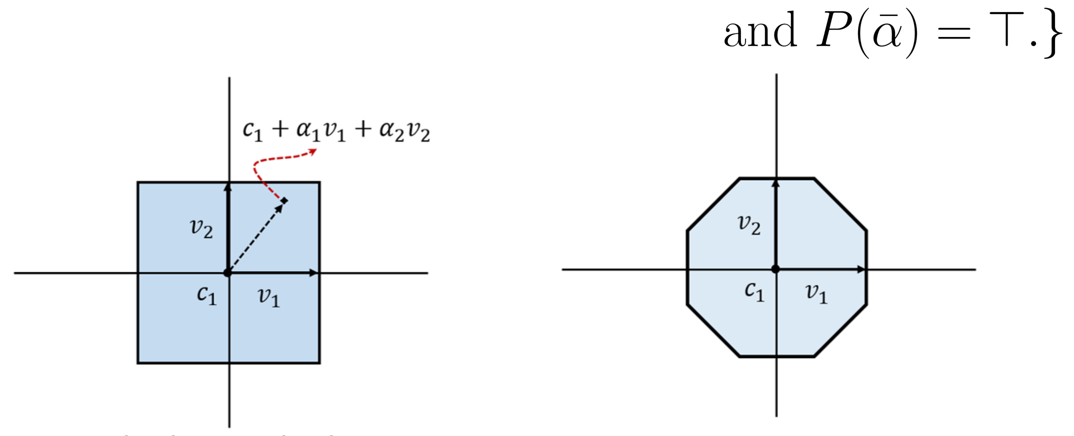


Figure: The state reached at time t from $x_0+\alpha_1 v_1+\alpha_2 v_2$ is identical to $\tau_i(x_0,t)+\alpha_1(\tau_i(x_0+v_1,t)-\tau_i(x_0,t))+\alpha_2(\tau_i(x_0+v_2,t)-\tau_i(x_0,t))$.

Generalized Star

A set is represented as $\Theta = \langle c, V, P\rangle$ where $c \in \mathbb{R}^n$, $V = \{v_1, \dots, v_m\}, P : \mathbb{R}^m \rightarrow \{\top, \bot\}$ where.

$$[\![\Theta]\!] = \{x \mid \exists \bar{\alpha} = [\alpha_1, \dots, \alpha_m]^T, x = c + \Sigma_{i=1}^n \alpha_i v_i \text{ and } P(\bar{\alpha}) = \top.\}$$



$P \triangleq |\alpha_1| \le 1 \wedge |\alpha_2| \le 1$ $P \triangleq |\alpha_1| \le 1 \wedge |\alpha_2| \le 1 \wedge |\alpha_1+\alpha_2| \le 1.5$

Figure: Examples of sets in generalized star representation.

Rechable Set Computation

Reachable set $Reach(\langle c, V, P\rangle, t) = \langle c', V', P\rangle$ where $c' = \tau(c,t)$ and $V' = \{v'_1, v'_2, \dots, v'_m\}$ where $v'_i = \tau(c+v_i, t) - \tau(c,t)$.

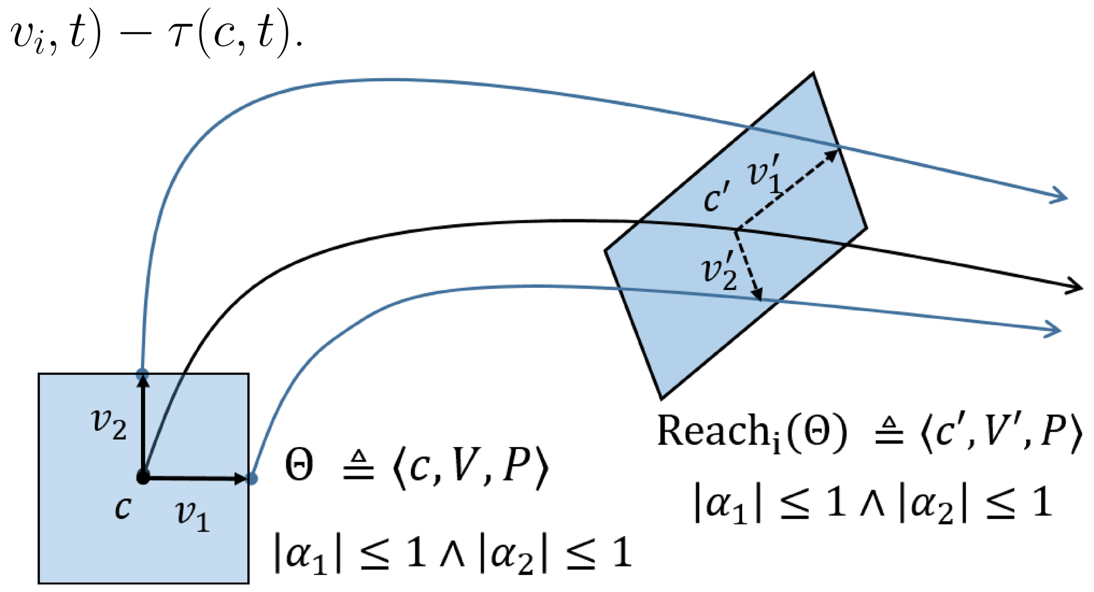


Figure: Illustration of the reachable set computation using simulations and generalized star representation.

Constraint Propagation for Invariants

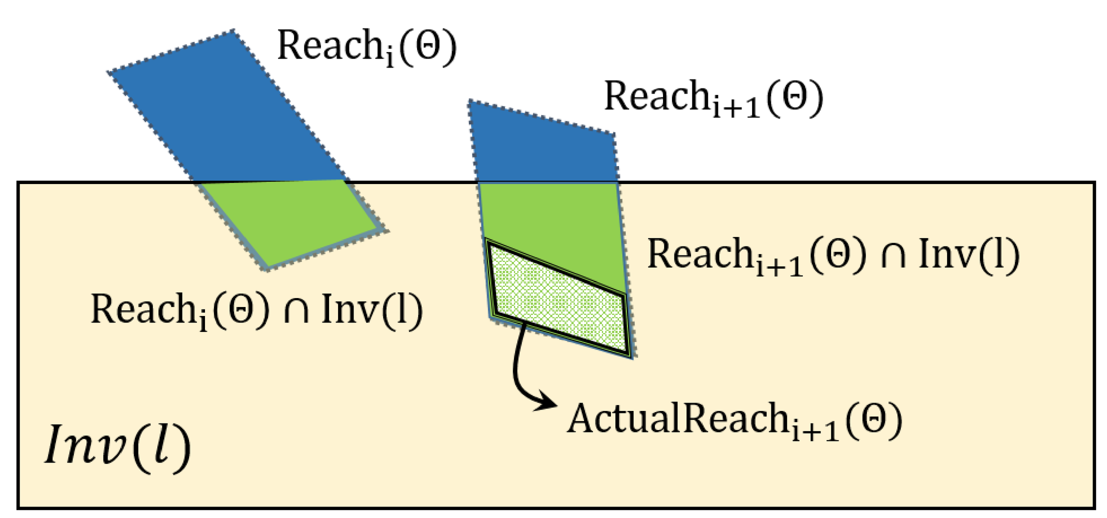


Figure: Figure depicting the overapproximation of the reachable set computed by performing $Reach_j \cap Inv(l)$.

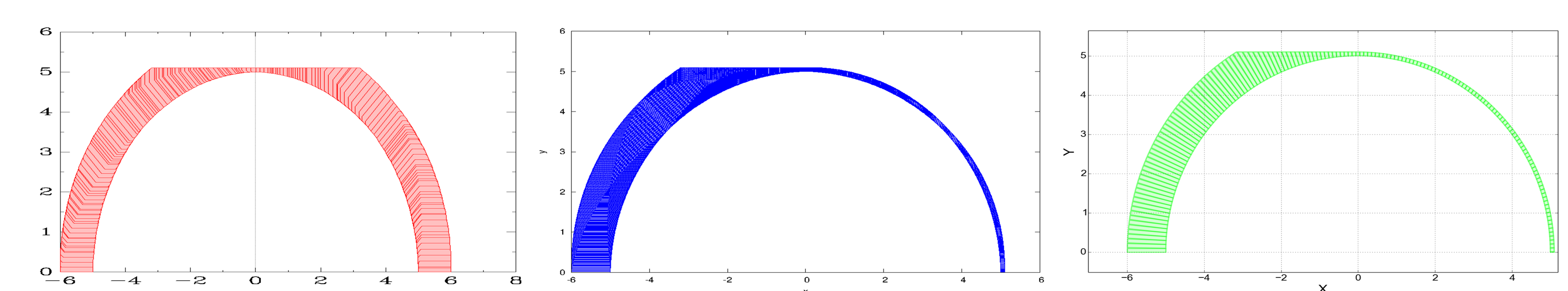
Solution: Given $Reach_j = \langle c_j, V_j, P\rangle$, convert $Inv(l)$ to a star representation as $Inv = \langle c_j, V_j, Q_j\rangle$ and add the constraints Q_j to all the future predicates from j as $Reach'_j = \langle c_j, V_j, P \wedge Q_j\rangle$.

Aggregation Techniques for Discrete Transitions

Reachable set from an initial state Θ would have multiple sets that encounter a discrete transitions, say k. The number of sets to track after d number of discrete transitions would grow exponentially as k^d. Aggregating all the sets that take a discrete transition into one set would lead to an overapproximation that is too conservative. Not aggregating would lead to increase in the time taken for verification.

Solution: Perform need based aggregation. Successors $S_1, S_2, \dots, S_k$ are all aggregated as S_{agg} by default. If the reachable set from S_{agg} reaches a *guard* state for a discrete transition, it is de-aggregated and only the sets $S_{j_1}, S_{j_2}, \dots, S_{j_r}$ which have at least one concrete transition are aggregated.

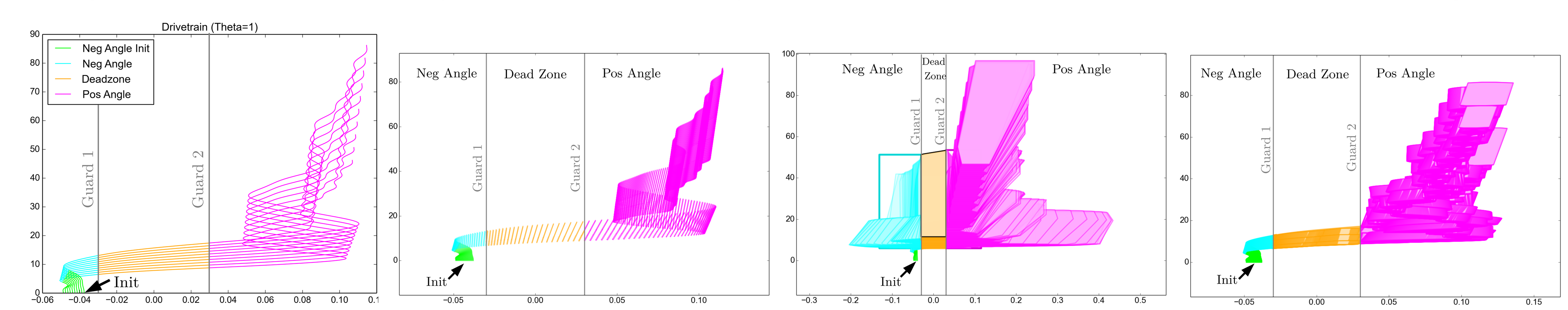
Results



(a) SpaceEx stc (b) Flow* (c) HyLAA

Step	No Trim	Trim
0.05	16	5
0.005	119	9
0.001	576	25
0.0005	1148	45

Figure: The harmonic oscillator system with invariant $0 \le y \le 5.1$ demonstrates the benefit of invariant constraint propagation.



(a) Simulations (b) Unaggregated (c) Aggregated (incomplete) (d) Deaggregated

Figure: Projections of x_3 versus x_1 for the 10-dimensional drivetrain system. While complete aggregation fails to complete for this model, using deaggregation produces a similar plot to the unaggregated method in less time.

Conclusion

HyLAA implements a dynamic analysis technique for computing simulation-equivalent reachable set for linear hybrid automata. Invariant constraint propagation and on-the-fly de-aggregation techniuqes improve the efficiency of the implementation while providing soundness and relative-completeness guarantees.

References

[1] Parasara Sridhar Duggirala and Mahesh Viswanathan. Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*, pages 477–494. Springer, 2016.

[2] Stanley Bak and Parasara Sridhar Duggirala. Rigorous simulation-based analysis of linear hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2017.