Reducing the Wrapping Effect in Flowpipe Construction using Pseudo-Invariants (Work in Progress)

Stanley Bak
stanleybak@gmail.com
United States Air Force Research Lab - Information Directorate - Rome, NY, USA

ABSTRACT

The reachability problem for a nonlinear hybrid automaton is often decomposed into steps where continuous successors are computed, and steps where discrete transitions and reset maps are processed. In this paper, we will show one method which can reduce the wrapping effect in the continuous-successor computation stage. A reduction in the wrapping effect can lead to both reduced error and reduced computation time.

The key insight behind the proposed method is that, when computing continuous successors, time need not be tracked precisely. We split an individual mode of a hybrid automaton into a pair of modes by introducing an artificial invariant (and associated transition) which we call a *pseudo-invariant*. The resultant hybrid automaton is a bisimulation of the original one, and thus their exact set of reachable states is identical. However, since time information is often dropped across discrete transitions, practical methods for overapproximating reachability can experience less wrapping-effect error when run on the constructed bisimulation. We demonstrate the advantage of the approach of pseudo-invariants by computing reachability for a nonlinear dynamical system using Flow*, a state-of-the-art reachability tool.

1. INTRODUCTION

Consider a hybrid automaton [1] where the dynamics in each mode are defined by nonlinear differential equations. Computing the continuous successors of the hybrid automaton in a single mode consists of solving an initial value problem restricted to the mode's invariant:

$$y'(t) = f(y),\ y(0) = s_0,\ y \in \mathbb{R}^n,\ t \in \mathbb{R}$$

For nonlinear dynamics, many methods for directly computing continuous post images rely on a process called flowpipe construction. This process iteratively computes the set of states reachable after some intermediate amount of time has elapsed. These methods start with an initial set of states s_0 at time 0, and then compute an image of the set of states after some positive amount of time, t_1, has elapsed. The output of this step, s_1 is then used as the input for the next step, with a possibly different time step, which produces an image of the reachable states s_2 at some later time t_2, and so on. Combined with method-specific reasoning about the states which can be reached between time steps, this approach can be used for computing time-bounded reachability, or, if the output states leave the mode's invariant, the continuous successors of a single discrete mode of a hybrid automaton.

This general approach is used in a wide range of analysis techniques and tools for hybrid systems, each with different representations and trade-offs. Some of the various representations which have been considered include general polyhedra [2], support functions [3], orthogonal polyhedra [4], sets of hyperrectangles [5], and Taylor models [6].

One potential problem with these methods is the wrapping effect [7], which occurs when the representation used cannot exactly represent the state set at each step in the computation. In these cases, an over-approximation is used which is representable, and thus is still useful for proving safety properties. However, since the input to each step depends on the output of the previous step, this error can accumulate and may blow up.

In some classes of systems, representations can be used which can always exactly represent the output at each step. For example, in linear systems, the reachability operator is described by an affine transformation, so if the input set is a polytope, the output set will also be a polytope. For systems with general nonlinear dynamics, however, there is no known exact and concise representation for all possible reach sets. In this case, an over-approximation step becomes necessary, and the accumulated error problem due to the wrapping effect becomes a legitimate concern.

In this paper, we present a technique for reducing the wrapping effect. The key insight is that, for the purposes of hybrid systems reachability, if dynamics are described by time-invariant, autonomous ODEs, *time need not be tracked so precisely.*

In the next section, we introduce some preliminary definitions. Section 3 then discusses the paper's main contribution, the technique of pseudo-invariants. This technique works by splitting a single mode of a hybrid automaton into two modes, with a specially-crafted condition on when the switching should occur, called the pseudo-invariant. The reachable set of states for the two automata is shown to be bisimilar. However, since hybrid systems reachability tools commonly aggregate sets across discrete transitions, the in-

CyPhy' 14 Berlin, Germany
Copyright 2014 ACM 978-1-4503-2871-5/14/04
http://dx.doi.org/10.1145/2593458.2593471 ...$15.00.

serted transition serves to reduce wrapping-effect error. In Section 4, we demonstrate that pseudo-invariants can benefit state-of-the-art reachability tools by applying it to the Flow* tool [8] on a nonlinear system. Finally, conclusions are given in Section 5.

2. PRELIMINARIES

In this section, we provide preliminary definitions used throughout the rest of the paper.

A **hybrid automaton** consists of a set of discrete modes, a set of n real-valued continuous variables, and a set of transitions. Each discrete mode has associated with it an **invariant** and a **dynamics function**. The invariant is a predicate on the continuous variables which restricts the situations when time can elapse inside the discrete mode. The dynamics function defines differential equations which described the evolution of the continuous variables as time is elapsing. This function is assumed to be defined and Lipschitz continuous, thus the solution to the initial value problem is defined and unique. Each transition contains a **source** mode, and a **destination** mode, as well as a **guard** and a **reset**. The guard is a predicate on the continuous states which indicates when the transition is enabled. The reset mapping is a function from the continuous variables to the continuous variables, which is applied when the transition is taken. When we omit explicit mentioning a transition's reset, the identity reset is implied.

A **continuous successor** from a state (M, $\mathbf{x}$) is a state (M, $\mathbf{x}$') where **x'** is a solution to the initial value problem for the dynamics defined in mode m for some positive amount of time starting at **x**, and that solution remains within the invariant of mode M for the time interval being considered. A **discrete successor** from a state (M, $\mathbf{x}$) is a state ($M^{'}$, **x**') if there is a transition with source M, destination $M^{'}$, where the guard predicate evaluates **x** to true, and the transition reset maps **x** to **x**'. A **trajectory** from an initial state (M, $\mathbf{x}$) is a (possibly infinite) list of states, such that the first state is (M, $\mathbf{x}$), and each subsequent state is either a continuous or discrete successor of the previous state in the list.

The **reachability** of a hybrid automaton from a set of initial states is the set of states which are in any trajectory from a state contained in the initial set. **Time-bounded reachability** is equivalent to reachability, with the condition that the sum of the times in the continuous successors of the trajectory is less than some bound.

Many ways have been investigated in order to compute reachability, depending on the restrictions placed on the hybrid automaton. For general hybrid automata, computing reachability is undecidable [9], so typically an overapproximation is computed which is sufficient for verifying safety properties. Here, we concern ourself with methods which explore the set of states starting from the initial set. These methods rely on quick and accurate ways to compute (or overapproximate) discrete and continuous successors from a set of states. The primary focus of this work is decreasing wrapping-effect error in the computation of continuous successors.

3. PSEUDO-INVARIANTS

In the method of pseudo-invariants, a single mode of the original hybrid automaton M is split into two modes, M_1 and M_2. This is similar to the splitting done in the linear phase-portrait approximation method [10], except that we do not change the dynamics and can preserve a bisimulation relationship.

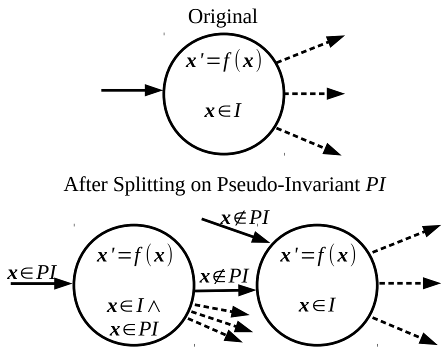


Figure 1: In the method of pseudo-invariants, a single mode of the hybrid automaton is split into two.

The splitting is done based using a condition, $x \in PI$, called the *pseudo-invariant*. Specifically, the dynamics, invariants, and outgoing discrete transitions of M are copied to M_1 and M_2. The incoming transitions to the original M are copied. One copy is directed to M_1 with the additional guard condition $x \in PI$, the other copy is directed to M_2 with the extra guard condition $x \notin PI$. An additional invariant, the pseudo-invariant, is added to M_1. A transition is also added from M_1 to M_2 with the guard condition $x \notin PI$.

The construction is shown in Figure 1 for a part of a generic hybrid automaton. Here, the invariant $x \in PI$ is used to split the mode at the top into the two modes shown at the bottom (M_1 is on the left and M_2 on the right). The resulting automaton is a bisimulation of the original one, meaning that all executions of the original automaton have a corresponding trajectory in the constructed automaton, and vice-versa. This can be shown in two parts.

First, for every trajectory in the first automaton, if there is continuous successor related to the mode being split, there are three possibilities: either the solutions of the ODE start and stay in PI before taking an outgoing discrete transition, or they do not start in PI, or they start in PI and then at some time leave PI before the outgoing discrete transition. The first case can be simulated by the bottom automaton by entering the left mode M_1, then exiting through a copied outgoing transition in M_1. The second case can be simulated by entering through a transition directly to M_2, and then proceeding as if M_2 were the original mode M. The third case can be simulated by remaining in M_1 until the solution leaves PI at which point we transition to M_2 and finish the corresponding continuous successor. This third case may reduce wrapping error, as we will soon show.

In the other direction, we need to show that no new behaviors are introduced by the splitting process. This can be shown, again, by case analysis of the possible successors in the bottom automaton, and is omitted here.

Intuitively, the bottom automaton may reduce wrapping because computational methods will typically cluster or aggregate states when crossing a discrete transition. For example, SpaceEx [3] can use a convex hull to perform clustering before time is advanced in successor modes. Flow* [11] can

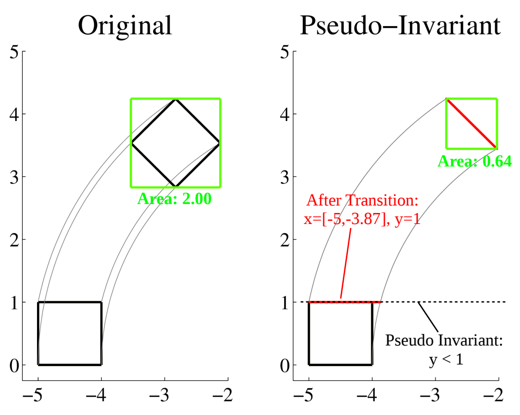


Figure 2: The method of pseudo-invariants reduces wrapping error.

perform interval or parallelotope aggregation after a discrete transition. With the method of pseudo-invariants, a bisimulation is constructed with a transition that can force this aggregation to occur. This can, in certain cases, reduce both computation time as well as error from the wrapping effect. Notice that even if time is not precisely tracked across transition boundaries, it is irrelevant when computing (time-unbounded) reachability, and continuous successors.

Example

Consider the prototypical example of the wrapping effect, the harmonic oscillator ($\dot{x} = y$, $\dot{y} = -x$) where the representation for sets of continuous states is intervals aligned with the axes, as shown in Figure 2. On the left, reachability from $S_0 = ([-5,-4] \times [0,1])$ is computed using a step of $t_1 = \frac{\pi}{4}$, and then the result is over-approximated with intervals aligned to the axes, which gives an S_1 (the green box) of area 2.0.

On the right, the method of pseudo-invariants is used where the inserted pseudo-invariant is $y < 1$. Manually, we can compute the intersection between the line $y = 1$ and the reach set from $S_0 = ([-5,-4] \times [0,1])$. For the harmonic oscillator, the solution is $x(t) = y(0) * sin(t) + x(0) * cos(t)$ and $y(t) = y(0) * cos(t) - x(0) * sin(t)$, so the initial point $(-4, 0)$ reaches $y = 1$ at $t = sin^{-1}(\frac{1}{4})$ and $x = -4 * cos(sin^{-1}(\frac{1}{4})) = -3.87$. The intersection of the reach set with the pseudo-invariant guard $y \geq 1$ is therefore $([-5,-3.87] \times [1,1])$. Using $([-5,-3.87] \times [1,1])$ as an initial set and computing a single continuous step with $t_1 = \frac{\pi}{4}$, the resultant over-approximation S_1 has an area of 0.64 (Figure 2, right), a significant decrease in wrapping error.

Practical Considerations. When using pseudo-invariants inside a tool, the intersection computed for the transition may be an overapproximation of the exact intersection, but this can, nonetheless, still reduce representation error and therefore reduce the wrapping effect. The actual invariant that is used also has an effect on the error (for example, using $y < 2$ in the harmonic oscillator would result in a different over-approximation error), as well as the set representation used by the underlying method.

Additionally, notice that not all trajectories are guaranteed to leave the pseudo-invariant set. This does not affect correctness, since, as in the first case of the bisimulation

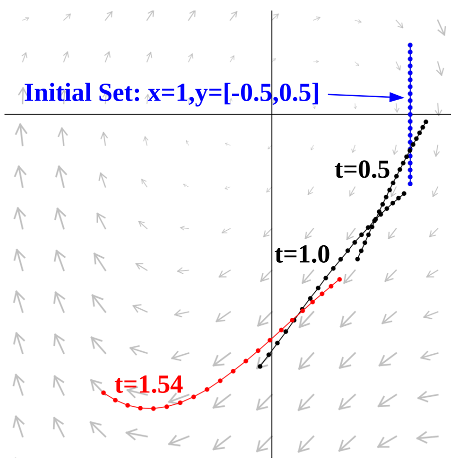


Figure 3: Simulating the Van der Pol system shows that, at the time Flow* errors (t=1.54, red), the reach set is large and nonlinear.

discussion, the trajectories can still proceed to subsequent modes. The effect of the pseudo-invariant in these cases is just that wrapping error is not reduced beyond that in the original automaton.

4. REDUCING WRAPPING IN FLOW*

In this section, we show that the method of pseudo-invariants can benefit state-of-the-art tools when computing reachability for nontrivial systems. Specifically, we use the Flow* tool [6, 8] which performs reachability using integration of Taylor models, on the Van der Pol oscillator system with the following dynamics:

$$\begin{aligned} x' &= y \\ y' &= (1 - x^2) * y - x \end{aligned}$$

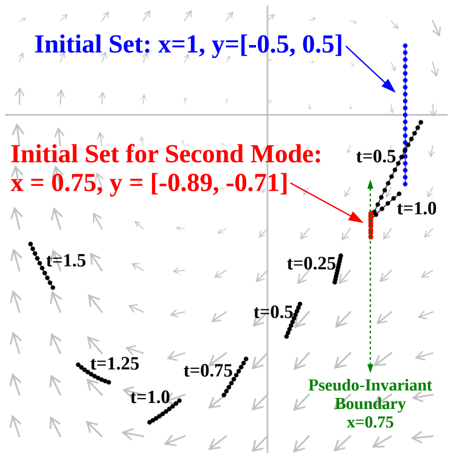
We used an initial set of $x = 1$, $y = [-0.5, 0.5]$ and the following Flow* parameters: reachability time, 10; step size, 0.1; remainder estimation, 1e-9; preconditioning, QR; adaptive orders, [4,15]; cutoff 1e-15; precision, 53; successor aggregation, parallelotope.

The tool was run on the system, computing reachability. After about 60 seconds of wall time, however, Flow* begins running extremely slow and finally returns the error that the remainder estimation is not large enough and it cannot proceed past reachability time 1.54.

The reason for this can be seen by simulating the system, as shown in Figure 3. At time 1.54, the true reach set becomes both large and nonlinear. Although Flow*'s reach set representation, Taylor models, can handle some level of nonlinearity, in this case the large and nonlinear reach set cannot be represented accurately enough, and wrapping-effect error accumulates until the tool exits with an error.

Next, we augmented the Van der Pol system with the pseudo-invariant $x > 0.75$. In this case, we expect the computation to proceed first by computing reachability restricted to points where $x > 0.75$, then the discrete transition to be taken and states reaggregated along $x = 0.75$, after which computation will proceed in the second mode of the pseudo-invariant construction.

A simulation of the expected reach sets is shown in Figure 4. In the figure, times shown to the right of the pseudo-



Initial Set: x=1, y=[-0.5, 0.5]
Initial Set for Second Mode:
x = 0.75, y = [-0.89, -0.71]
t=0.5
t=1.0
t=1.5
t=0.25
t=0.5
t=1.25
t=0.75
t=1.0
Pseudo-Invariant Boundary x=0.75

Figure 4: Along the pseudo-invariant boundary $x = 0.75$ (green dotted line), the reach set is aggregated and the subsequent intermediate reach sets are smaller and have less nonlinearity.

invariant boundary (green dotted line) are relative to the start of the computation in the first mode of the pseudo-invariant split. Times to the left of the pseudo-invariant boundary are relative to the start of the continuous successor computation in the second mode. Notice that, after the discrete transition, the reach set time-projections are no longer at fixed times relative to the initial set. However, the reachable set of states over all time is the same. This shows that constructed bisimulation has the same reach set, although the error to represent the time-projection has changed. In this case, the time-projections are both smaller and have less nonlinearity, so we expect less wrapping-effect error.

This is indeed the case when we run the reachability computation on the pseudo-invariant system in Flow*. After about 100 seconds of wall time, Flow* successfully completes the reachability computation of the system. The result is shown in Figure 5. Although pseudo-invariants were only applied once, the method can be applied multiple times if further wrapping-effect error reductions are needed.

5. CONCLUSION

In this work, we presented the method of pseudo-invariants for reducing wrapping-effect error in flowpipe construction. This is typically the most costly step when computing the reachability of a hybrid automaton with non-trivial continuous dynamics, and the developed technique can reduce both computation time and error. Pseudo-Invariants work by splitting a single discrete mode into two modes with identical dynamics, in order to force tools to perform a state-set aggregation during the discrete transition. Pseudo-invariants are generic and can be applied to any algorithm with wrapping-effect errors. To demonstrate this, we showed its advantage in the computation of reachability for the Van der Pol system using Flow*, a state-of-the-art reachability tool.

The next step we plan to take is to automate the process of constructing the pseudo-invariants. One approach we are pursuing is to detect when wrapping-effect error is starting to get large, and then generating a pseudo-invariant on-the-fly. This can be done, for example, by taking the bounding box of the current time-projection of the reach set.

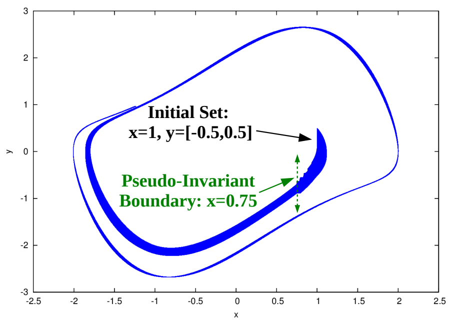


Figure 5: After inserting the pseudo-invariant, Flow* successfully computes the reach set of the Van der Pol system.

6. REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *THEORETICAL COMPUTER SCIENCE*, vol. 138, pp. 3–34, 1995.

[2] A. Chutinan and B. H. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations." Springer, 1999, pp. 76–90.

[3] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, ser. LNCS, S. Q. Ganesh Gopalakrishnan, Ed. Springer, 2011.

[4] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems: Computation and Control HSCC'98*. Springer, 1998, pp. 96–109, lNCS 1386.

[5] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo, "Sandboxing controllers for cyber-physical systems," in *Proceedings of International Conference on Cyber-physical systems (ICCPS 2011)*, 2011.

[6] X. Chen, E. Abraham, and S. Sankaranarayanan, "Taylor model flowpipe construction for non-linear hybrid systems," in *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, 2012, pp. 183–192.

[7] R. E. Moore, *Interval analysis*. Prentice-Hall.

[8] X. Chen, E. Abraham, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification (CAV)*, 2013.

[9] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Journal of Computer and System Sciences*. ACM Press, 1995, pp. 373–382.

[10] T. A. Henzinger, P.-H. Ho, and H. Wong-toi, "Algorithmic analysis of nonlinear hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 225–238, 1996.

[11] X. Chen, "User's manual of flow v1.2.0," 2013. [Online]. Available: http://www-i2.informatik.rwth-aachen.de/i2/fileadmin/user_upload/documents/HybridSystemsGroup/chen/manual-1.2.0.pdf