Reachability Analysis of Hybrid Systems: An Experience Report

Manish Goyal

***Abstract*—Hybrid systems are mathematical models of control systems whose safety verification is critical for many applications. In practice, a rigorous tool is still not available for verifying every class of hybrid systems. HyTech was the first attempt in this direction followed by PHaver, both restricted to Linear Hybrid Automata (LHA). HSolver is another successful contribution for verification of nonlinear systems. PHaver can efficiently verify safety properties with the help of piecewise constant bounds on derivatives. Its use is greatly motivated by on-the-fly overapproximations of piecewise affine dynamics with various user-specified parameters. HSolver verifies safety of nonlinear systems using constraint propagation based abstraction refinement. We have evaluated a few examples and shown that both tools have their strengths and weaknesses. In all the examples, the approximation of nonlinear systems by linear systems is performed by the rate translation.**

***Keywords*—Hybrid systems, reachability analysis, PHAVer, HSolver, linear approximation, rate translation.**

I. Introduction

HYbrid systems are combinations of discrete as well as continuous dynamics and are analyzed using techniques from computer science and control theory. Over the years, these systems have proved their significance in safety critical applications. However, the safety verification has always been a challenge because of their complex behavior. This has prompted researchers to seek efficient methods to verify subclasses such as linear hybrid systems [ACH+95] and nonlinear hybrid systems.

HyTech was the first reachability analysis tool developed by Henzinger et al. [HHWT97] for linear hybrid systems. It was featured with a powerful input language but limited by the overflow errors due to the restricted number of digits. Being a first implementation in this direction, HyTech was more of a prototype based on which more powerful practical tools were developed. HyTech had led the researchers to improve its underlying algorithm by various abstraction techniques.

In recent years, the research on reachability analysis of hybrid systems has gained a new impetus with the development of various tools. For e.g., PHaver [Fre05] and HSolver [RS05], both exhibit altogether different methodologies for the hybrid systems safety verification. Their applicability in numerous real world scenerios has emphasized that a broadening of the practical utility of these tools has been achieved.

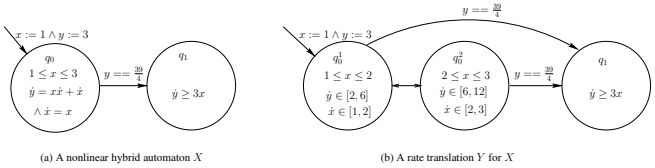
The objective of this paper is to compare the two tools with respect to various parameters. We have evaluated many benchmarks with slight modifications, if required, to understand the limits of their applicability, i.e., their strengths and weaknesses. For this purpose, we have presented experimental results based on the analysis of various paradigmatic examples. PHAVer can only verify systems with affine linear dynamics. Therefore, *rate translation* [HWT95] technique has been used to approximate a nonlinear system by its linear counterpart.

The paper is structured as follows. Section II presents the related work in this area. Sections III states few definitions besides the tools descriptions which are essentially from the papers [Fre05] and [RS05]. In Section IV, we have described the benchmarks followed by their experimental outcomes in Section V. These numerical results help in better assessment of the tools. We have compared the features of the tools in Section VI and finally, a few conclusions about the tools behavior have been presented in Section VII.

II. Related Work

The need for a rigorous hybrid system verification tool prompts the researchers to discuss the characteristics of existing tools which in turn renders the further growth in this direction. Ben Makhlouf et al [MS06] have evaluated the tools PHAVer and HSolver. They have assessed several benchmarks to explain the tool behaviors in terms of time and memory consumed. They concluded that PHAVer runs faster than HSolver in linear hybrid systems. On the other hand, HSolver is fast for the verification of nonlinear systems. But it was difficult to draw a single conclusion in terms of the memory consumption. Carloni et al [CPPSV06] have worked on the similar line and analyzed several hybrid system tools. The authors compared the tools in terms of their syntax and semantics, design aspects, capabilities and their solution methodologies. For comparison, they have divided the tools in two broad categories, *simulation centric* and *formal verification centric*. The authors pointed out the need for the unification of the design paradigms of various hybrid systems tools because limited by their languages, syntax and assumptions, it gets difficult to share information among various tools. They have suggested a *semantic-aware interchange format* based on the abstract semantics which facilitates the import and export of the design specifications and thus making a formal comparison between the tools easier. Our work is a further step in the evaluation of the tools PHAVer and HSolver as we have focused on exploring more features. PHAVer allows the users not only to control the verification but also to compute the simulation relation, compositional reasoning, parametric analysis, to choose among search strategies. Similarly, HSolver lets the user to specify the number of abstract states. Also, the *rate translation* helped us to compute the linearization error which is otherwise difficult to calculate.

Indian Institute of Technology Guwahati, India - 781039 emails: g.manish@iitg.ernet.in, manish.lucky@gmail.com



(a) A nonlinear hybrid automaton X

(b) A rate translation Y for X

Fig. 1. The rate translation

III. Preliminaries

Def. *(Hybrid Automaton)* [Joh00] - A hybrid automaton is a tuple $H = (Loc, Var, f, Init, Inv, Jump, \Sigma)$ whose components are:

- A finite set Loc of discrete modes. It represents the discrete dynamics of H.
- A finite set $Var = \{x_1, x_2, ..., x_n\}$ of real valued continuous variables x_i, $1 \le i \le n$, $n \ge 0$. It represents the continuous dynamics.
- A function $f : Loc \times \mathbb{R}^n \to \mathbb{R}^n$ is called the vector field. It defines the continuous flow in each discrete mode $l \in Loc$ through a differential equation $\dot{x} = f(l, x)$, $l \in Loc, x \in Var$.
- The initial condition is a set $Init \subseteq Loc \times \mathbb{R}^n$ that defines the initial state of H. A state of hybrid automaton is a pair (l, v) that consists of a discrete mode $l \in Loc$ and a point $v \in \mathbb{R}^n$ being a valuation over Var where a valuation is a function $val : Var \to \mathbb{R}$.
- The set $Inv \subseteq Loc \times \mathbb{R}^n$ called the invariant condition. As long as the H is in discrete mode $l \in Loc$, the state must belong to Inv.
- A set-valued function $Jump : Loc \times \mathbb{R}^n \to P(Loc \times \mathbb{R}^n)$ called the jump condition.
- A finite set Σ of events where each jump is represented by an event.

Def. *(Hybrid I/O automaton)* [LSV01] - A hybrid Input/Output automaton (HIOA) is a hybrid automaton such that

- Var is a finite and disjoint set of state and input variables, Var_S and Var_I, and of output variables $Var_O \subseteq Var_S$ where $Var = Var_S \cup Var_I$.

Def. *(Linear and Affine Hybrid Automaton)* [Hen96] - A *linear hybrid automaton* (LHA) is a hybrid automaton in which the invariants and the jumps are given by linear formulas over Var_a, and the flows are given by linear formulas over $\dot{Var}_a$. A *linear formula* is a finite disjunction of convex linear formulas and, a *convex linear formula* is a finite conjunction of constraints $\Sigma_i a_i x_i + b \bowtie 0$, with $a_i, b \in \mathbb{Z}$, $x_i \in Var$ and $\bowtie \in \{<, \le, =\}$ over the linear expressions $\Sigma_i a_i x_i + b$. Whereas, if the dynamics are given by *linear formulas* over the derivatives and the variables, then it is called as affine hybrid automaton (AHA).

Def. *(Rate Translation)* [HWT95] - The rate translation approximates a nonlinear hybrid automaton by a linear hybrid automaton. It consists of two steps:

- Partitioning the state space within each location
- Replacing nonlinear dynamics within each region of the partitioned state space by piecewise-constant bounds on derivatives.

It is generally assumed that all invariants, initial and jump conditions of the given nonlinear hybrid automaton are convex linear predicates.

Example 1. Consider the hybrid automaton X in Fig. 1. The flow condition for location q_0 is $\dot{x} = x \wedge \dot{y} = x\dot{x} + \dot{x}$. The solution of the flow equation in q_0 is of the form $y = \frac{1}{2}x^2 + x + k$. From the initial valuation $(1, 3)$, the valuation $(3, 9)$ in location q_0 is reachable but the location q_1 is not. The flow vectors in location q_0 are of the form $(x, x^2 + x)$, $\forall\ 1 \le x \le 3$. Now, we carry out the rate translation Y of X with the partitioning that splits the invariant in q_0 along the line $x = 2$ into two locations q_0^1 and q_0^2. Since $\dot{x}$ may vary from 1 to 2 in the location q_0^1, the flow condition in Y for q_0^1 is given by $\dot{x} \in [1, 2] \wedge \dot{y} \in [2, 6]$. The flow condition for q_0^2 is $\dot{x} \in [2, 3] \wedge \dot{y} \in [6, 12]$ and therefore, q_1 is reachable in Y.

A. PHAVer

PHAVer (Polyhedral Hybrid Automaton Verifyer) [Fre05] is a tool for the safety verification of Linear Hybrid Automata (LHA) which can be analyzed using polyhedra, i.e., finite convex linear formulas. It makes use of a general framework of Hybrid I/O automata with affine dynamics. As PHAVer's computations are based on LHA, it conservatively over-approximates affine dynamics by linear dynamics. However, the over-approximation error depends on the location size and the dynamics and so the tool provides the functionality to partition the locations along a suitable hyperplane during analysis until a minimum threshold is reached. The Parma Polyhedra Library (PPL) achieves robust and infinite precision arithmetic. The algorithm computes the set of states reachable from an initial state. An expert user can control the location refinement by combining and prioritizing various parameters. Moreover, the tool provides the user with the liberty of controlling the bits, constraints and iterations. Various abstractions like convex-hull and bounding-box used for simplification of polyhedra results in the forced termination of the algorithm in few cases. PHAVer also supports compositional reasoning.

B. HSolver

HSolver, a safety verification tool for nonlinear hybrid systems, was developed by Stephan et al. [RS05] based on a package RSolver [Rat06] that provides pruning and solving of quantified constraints of the real numbers. In this tool, the state space is divided into rectangular grids and, interval arithmetic is used to check the trajectories on the boundary of neighboring grid elements. The approach is used in the abstraction refinement framework where piecewise splitting of abstract states as hyper-rectangles (boxes) is performed until a fixed point is reached and transitions are recomputed giving us a abstract discrete system. The safety of this abstract system implies the safety of the original hybrid system. However, in order to avoid an exponential splitting, an interval constraint propagation based refinement step is employed. The beauty of this method is that it allows jump conditions, initial states and unsafe states to be described by complex constraints and then pruning algorithm is used to remove the elements that do not satisfy these constraints from the boxes.

IV. BENCHMARKS

A. Damping Pendulum

Consider a pendulum hanging from a weight-less solid rod and moving under gravity [Lyg04]. Let θ denotes the angle the pendulum makes with the downward vertical, l the length of the pendulum, m its mass, and K the damping coefficient. The flow can be represented as the following equation:

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) - \frac{K}{m}x_2 \end{bmatrix},$$

where $x_1 = \theta$ and $x_2 = \dot{\theta}$

Therefore, our nonlinear system is described as

- Flow: $(\dot{x_1}, \dot{x_2}) = (x_2, -\frac{g}{l}\sin(x_1) - \frac{K}{m}x_2)$
- Empty Jump relation
- Init: $x_1 = 1.048 \wedge x_2 = 1$
- Unsafe: $x_2 \leq 0$
- State space: $[-1.048, 1.048] \times [0, 1.2]$

Linear Approximation: Its important to mention that $\sin(x_1)$ is almost equal to x_1 for very small values of x_1 i.e., $-1.048 \leq x_1 \leq 1.048$. However, to substitute $\sin(\theta)$ by θ, we need to have a scaling factor z to get this linearization compensated. We refine the rate over x_1 into few intervals so that in each interval the scaling factor makes this linear approximation as close as possible to the original system. Now, suppose $\theta_l \leq \theta \leq \theta_u$. We can now calculate the scaling factor $\theta/\sin(\theta)$ over few values in this given interval and then take an average of these. This mean serves as a scaling factor z for this location. Now, we can also calculate the % linearization error (δ) as

$$\delta = \frac{|\theta \div z - \sin(\theta)|}{\sin(\theta)} \times 100$$

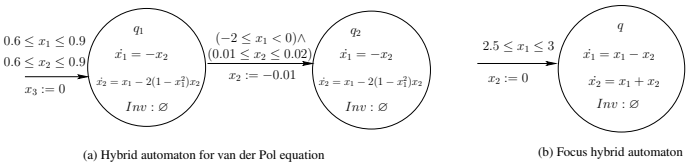
B. Train-gate-controller

The linear system consists of three components, the train, the gate, and the gate controller [AK04]. The train moves on a circular track of length. A road crosses the track and it is guarded by a gate which is controlled by a controller. The variable y represents the location of the train and x states the height of the gate. The composed system is described with s being the automaton state as

- Flow: $((s = 1 \vee s = 2 \vee s = 5) \rightarrow (5 \leq \dot{y} \leq 10 \wedge \dot{x} = \frac{1-x}{2})) \wedge ((s = 3 \vee s = 4) \rightarrow (5 \leq \dot{y} \leq 10 \wedge \dot{x} = \frac{10-x}{2}))$
- Jump: $((s = 1 \wedge y = 5) \rightarrow (s' = 2)) \vee ((s = 2 \wedge y = 15) \rightarrow (s' = 5)) \vee ((s = 2 \wedge y \leq 15) \rightarrow (s' = 3)) \vee ((s = 3 \wedge y = 15) \rightarrow (s' = 4)) \vee ((s = 4 \wedge y = 5) \rightarrow (s' = 3)) \vee ((s = 4 \wedge y \leq 5) \rightarrow (s' = 1)) \vee ((s = 5 \wedge y \leq 5) \rightarrow (s' = 1))$
- Init: $x = 1 \wedge y = 0$
- Unsafe: $x < 5 \wedge y = 0$
- State space: $[0, 25] \times [0, 10]$

C. Room Heating Benchmark

We consider a linear, three dimensional example of a room heating problem defined by 3 rooms and 2 heaters [FI04]. The objective is to maintain a minimum specified temperature in each and every room. And, if the temperature in a room falls below a threshold, then either heater is turned on if it is already present in the room or moved from the neighboring room. The temperature of a room depends on the difference of the temperature with the other rooms, the difference with the outside temperature, and on whether the heater is present and whether the heater is switched on/off. For the system description, refer [FI04].



(a) Hybrid automaton for van der Pol equation

(b) Focus hybrid automaton

Fig. 2. Hybrid automata for van der Pol and focus benchmarks

D. van der Pol Equation

We consider a modified 3-dimensional van der Pol second order equation with a time variable and some discrete jumps is used [RS05]. This hybrid automaton, shown in Fig 2(a), can be explained as

- Flow: $(\dot{x_1}, \dot{x_2}) = (-x_2, x_1 - 2(1 - x_1^2)x_2)$
- Jump: $((s = 1 \wedge -2 \leq x_1 < 0 \wedge 0.01 \leq x_2 \leq 0.02) \rightarrow (s' = 2 \wedge -0.01 \leq x_2 \leq -0.01 \wedge x_1' = x_1 \wedge x_3' = x_3))$
- Init: $0.6 \leq x_1 \leq 0.9 \wedge 0.6 \leq x_2 \leq 0.9 \wedge x_3 = 0$
- Unsafe: $(1 < x_1 \leq 2) \wedge (0.01 \leq x_2 \leq 2) \wedge (0 \leq x_3 \leq 6)$
- State space: $(1, [-2, 2] \times [0.01, 2] \times [0, 6]) \cup (2, [-2, 2] \times [-2, -0.01] \times [0, 6])$

Linear Approximation: The rate translation technique partitions the state space to approximate nonlinear dynamics. Here, we have divided the state space over x_1. After the partitioning, the nonlinear dynamics are overapproximated by linear flow. For example, the dynamics of the form $\dot{x_2} = x_1 - 2(1 - x_1^2)x_2$ in a location with state space $[0, 1] \times [0.01, 2]$ can be overapproximated by the dynamics of the form $x_1 - 2x_2 \leq \dot{x_2} \leq x_1$ using arithmetic equations. In this way, our original van der Pol system is approximated by a linear system keeping initial, unsafe states and jump relation in consideration.

E. Focus

A two dimensional system description adapted from [RS05] is shown in Fig. 2(b).

- Flow: $(\dot{x_1}, \dot{x_2}) = (x_1 - x_2, x_1 + x_2)$
- Empty Jump relation
- Init: $2.5 \leq x_1 \leq 3 \wedge x_2 = 0$
- Unsafe: $x_1 \leq 2$
- State space: $[0, 4] \times [0, 4]$

F. Billiards Game

A classical example of a linear system consisting of a billiards table of the dimensions $l = 13$ and $h = 10$ with a grey ball and a white ball [ACH+95]. Initially, the balls are placed at positions $p_g = (x_g, y_g)$ and $p_w = (x_w, y_w)$. The grey ball is kicked and moves with a constant velocity. The ball rebounds as soon as it reaches the table boundary. We

TABLE I
EXPERIMENTAL RESULTS ON PHAVER

PHAVer			
Benchmarks	Safety	Time(s)	Memory(KB)
Damping Pendulum	Unsafe Safe	0.10 0.25 (Refinement)	1328 2784
Train-gate-controller	Safe	0.63	4256
Room Heating	Unsafe	26.23 (Convex-Hull)	116672
van der Pol	Safe	0.15	2112
Billiards Game 1	Unsafe	0.70	3808
Billiards Game 2	Unsafe Safe	0.11 1.24 (Refinement & Convex-Hull)	1916 5984
Focus	Unsafe	0.11	1608

TABLE II
EXPERIMENTAL RESULTS ON HSOLVER

HSolver					
Benchmarks	Refine steps	Prune func #	Safety	Time(s)	Mem(KB)
Damping Pendulum	4	122	Safe	0.14	1920
Train-gate-controller	2	1675	Safe	1.50	3588
Room Heating*	14	248724	Unsafe	854.05	19720
van der Pol*	1	38	Safe	0.11	1212
Billiards Game 1	24	164718	Unknown	77.06	2600
Billiards Game 2	1	223	Safe	0.20	2404
Focus*	8	943	Safe	0.40	1880

have defined the following unsafe condition: (1) whether this grey ball hits the white ball, i.e., if the position p_w is reachable from the initial position p_g and (2) whether the ball crosses the table boundary.

- Flow: $(q_1 \rightarrow (\dot{x} = 2 \wedge \dot{y} = 1)) \wedge (q_2 \rightarrow (\dot{x} = -2 \wedge \dot{y} = 1)) \wedge (q_3 \rightarrow (\dot{x} = 2 \wedge \dot{y} = -1)) \wedge (q_4 \rightarrow (\dot{x} = -2 \wedge \dot{y} = -1))$
- Jump: The transition takes place as soon as the ball hits boundary of the table.
- Init: $x_g = 0 \wedge y_g = 0$
- UnSafe1: $x_w = 10 \wedge y_w = 8$
- Unsafe2: $x \geq 13 \wedge y \geq 10$
- State space: $[0, 13] \times [0, 10]$

V. EXPERIMENTAL OUTCOMES

.

All the benchmarks have been evaluated on a system with Intel Core 2 duo processor with 1.83 GHz clock and 2.0 GB of memory. We have used the *rate translation* for linearization as in the *damping pendulum* and *van der Pol* benchmarks. Of course this may not be a perfect approximation and the reachability results based on this transformation will be inconclusive with respect to the safety property because the linearization error has not been taken into account in most of the cases. Nevertheless, it helps us in comparing the functioning of the tools. We computed the maximum linearization error in the damping pendulum benchmark as 0.2%. Computational results are given in TABLE I. The damping pendulum system is declared as *unsafe* due to the over-approximation. However, we can still control this approximation by means of refinement which in turn limits the over-approximation and improves the precision. We can see that the damping pendulum as well as billiards game are proved to be safe after the application of refinement. PHAVer has the feature of compositional reasoning which makes it effective in a situation where more than one automaton interact. The train-gate-controller is one of this type of example. In the *room heating benchmark*, it took almost a minute to check the safety property with default polyhedra abstraction. However, the convex-hull (REACH_STOP_USE_CONVEX_HULL_ITER = 20) abstraction accelerated the termination and halved the verification time. Similarly, in van der pol benchmark, the bounding-box abstraction helped in the termination. PHAVer also provides the choices in the search strategies such as breadth-first and depth-first search. In van der Pol bencmark, we observed that depth-first search took more time for verification than breadth-first search. We have also noticed that limiting the number of bits and constraints speeds up the termination. For e.g. in the van der Pol example, changing the number of constraints threshold value from 72 to 48 reduced the time from 0.62 sec to 0.54 sec. However, the memory consumption by PHAVer is huge due to its use of polyhedral representation for the locations.

On the other hand, HSolver facilitates the verification of nonlinear hybrid systems. Experimental outcomes are tabularized in TABLE II. We can see that nonlinear dynamics are verified quite fast testifying to the tool's success in its use of pruning strategy which plays a significant role in the overall abstraction technique. HSolver verifies by removing points from the state space that are not on any trajectory from the initial to an unsafe state, and hence it solves the problem of excessive splitting. But in case of linear dynamics, it may not succeed which is clear from the billiards game benchmark. It takes a lot of time in the room heating benchmark. It was shown in the paper [RS05] that verification process for this example took > 10 hours. Therefore, we kept a limit of 100 on the number of abstract states to improve the termination.

VI. COMPARISON

If we compare these two tools in terms of verification time, we can see that PHAVer outperforms HSolver in some benchmarks. In systems where safety has been shown to be unknown (Billiards Game), HSolver requires a lot of time. Also, if a system consists of more than one automaton, HSolver requires a manual composition of these automata which is a tedious task. Whereas, PHAVer carries out the compostion of automata automatically and therefore, saves a lot of efforts. In terms of memory consumption, it is difficult to draw any single conclusion. In most of the cases, PHAVer has consumed more memory which results from its usage of polyhedra. But in cases where HSolver has exceeded PHAVer the refinement procedure could be held responsible. In the case of nonlinear systems, we can not conclude about their safety because the linearization error has not been taken into consideration except in the damping pendulum benchmark.
The strength of PHAVer lies in its feature of allowing the user to manipulate various parameters. The user can compute intersection as well as the difference of two state sets and can also perform parametric analysis. In the classical train-gate-controller example from [Hen96], we performed an existential quantification over the variables x, t and y using `reg = project_to(u)` where `reg` is the identifier

* The model is taken from [RS05].

for the set of states reachable in the automaton and `u` is the parameter which represents the reaction delay of the controller. PHAVer computed the values of `u` ($5\star u \geq 99$) for which the system is unsafe. However, use of HSolver is not leveraged by these functionalities. As the overall composed automaton consists of 27+ states, therefore, computing jump relations between these many states requires a lot of efforts.

PHAVer can also compute the simulation relation between two

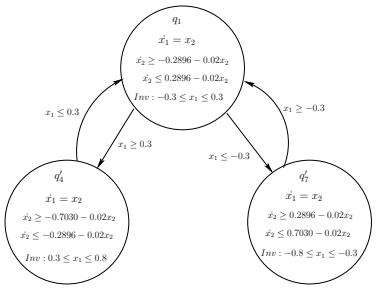


Fig. 3. An abstraction of the pendulum benchmark

automata. We computed a simulation relation using command `rel = get_sim(`H,H'`)` between original pendulum example (H) and its abstraction (H') where abstraction consists of fewer locations compared to the original benchmark. In original benchmark, the locations q_2, q_3 and q_4 have the dynamics $(-0.4698 - 0.02x_2) \leq \dot{x_2} \leq (-0.2896 - 0.02x_2)$, $(-0.6313 - 0.02x_2) \leq \dot{x_2} \leq (-0.4698 - 0.02x_2)$ and $(-0.7030 - 0.02x_2) \leq \dot{x_2} \leq (-0.6313 - 0.02x_2)$ with *invariants* $0.3 \leq x_1 \leq 0.5$, $0.5 \leq x_1 \leq 0.7$ and $0.7 \leq x_1 \leq 0.8$. These three locations are abstracted to q_4' with the bound $0.3 \leq x_1 \leq 0.8$ and the dynamics $(-0.7030 - 0.02x_2) \leq \dot{x_2} \leq (-0.2896 - 0.02x_2)$ by computing union over the bounds of locations q_2, q_3 and q_4 (cf. Figure 3). Similarly, the location q_7' is the abstraction of locations q_5, q_6 and q_7. When tested for safety, PHAVer deduced that the abstraction is safe too. The time taken and memory consumed are 0.20s and 2488KB as compared to the values 0.25s and 2784KB during verification of the original example. Although, there is no much improvement in terms of time but we can see a significant reduction in the memory requirements. The same trend was noticed in the focus benchmark.

VII. Conclusion

As expected, both tools have their limitations and strengths. PHAVer has proved its worth in linear hybrid systems verification. It helps the user to control the verification process by use of various parameters and abstraction techniques. It also provides discrete results in the examples where HSolver gave the results as *safety unknown*. Ben Makhlouf et al [MS06] and Carloni et al [CPPSV06] have also evaluated the tools. However, we have moved a step further by exploring the usage of more parameters provided by the tools. Our wide selection of the benchmarks with linear and nonlinear dynamics has helped us to better understand the tool behaviors in the diverse scenerios by using various features, e.g., in PHAVer simulation relation, parametric analysis, search strategy, composition of automata. The computation of the linearization error with the help of *rate translation* is another contribution. HSolver has a advantage over PHAVer as it can treat nonlinear dynamics better. Although it is not as rich as PHAVer in terms of features, it allows the user to customize the number of abstract states and the upper bound of the box diameter. The tools have taken a leap over existing reachability analysis methods. However, their deficiency in guaranteeing correct and timely results for every class of hybrid systems points to the need for future work in this direction.

References

[ACH+95] 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*, 138(1):3–34, 6 February 1995.

[AK04] Panos J. Antsaklis and Xenofon D. Koutsoukos. Hybrid systems: Review and recent progress. In Tariq Samad and Gary Balas, editors, *Software-Enabled Control*, pages 273–298. Institute of Electrical and Electronics Engineer, 2004.

[CPPSV06] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Sangiovanni-Vincentelli. Languages and tools for hybrid systems design. *Foundations and Trends in Electronic Design Automation*, 1(1/2), 2006.

[FI04] Ansgar Fehnker and Franjo Ivancic. Benchmarks for hybrid systems verification. In Rajeev Alur and George J. Pappas, editors, *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2004.

[Fre05] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In Manfred Morari and Lothar Thiele, editors, *HSCC*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2005.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the Eleventh Annual IEEE Symposium On Logic In Computer Science (LICS'96)*, pages 278–293, New York, USA, 1996. IEEE Computer Society Press.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. In Orna Grumberg, editor, *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–463. Springer, 1997.

[HWT95] Thomas A. Henzinger and Howard Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems*, volume 1066 of *Lecture Notes in Computer Science*, pages 377–388. Springer, 1995.

[Joh00] Karl H. Johansson. *Hybrid Systems Lecture Notes*. UC Berkeley, Spring 2000.

[LSV01] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Hybrid I/O automata revisited. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 403–417. Springer, 2001.

[Lyg04] J. Lygeros. *Lecture Notes on Hybrid Systems*. University of Patras, Greece, 2004.

[MS06] Ben Makhlouf and K. Stefan. An evaluation of two recent reachability analysis tools for hybrid systems. In *2nd IFAC Conference on Analysis and Design of Hybrid Systems*, 2006.

[Rat06] Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Log*, 7(4):723–748, 2006.

[RS05] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In M. Morari and L. Thiele, editors, *HSCC*, volume 3414 of *LNCS*, pages 573–589. Springer, 2005.

Acknowledgment

The author would like to thank his supervisor Dr. Purandar Bhaduri, friends Vallabh and Prabhat for their constructive feedback.