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 over approximations 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.**

***Index Terms*—HSolver, Hybrid systems, rate translation, reachability analysis, PHAVer.**

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 [1] and nonlinear hybrid systems.

HyTech was the first reachability analysis tool developed by Henzinger et al. [2] 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 [3] and HSolver [4], 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 [5] 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 [3], [4]. 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 [6] 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 [7] 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, and 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.

Manuscript received August 28, 2012; revised September 28, 2012.
The author is with Indian Institute of Technology Guwahati, India. (e-mail: g.manish@alumni.iitg.ernet.in).

III. Preliminaries

Definition (*Hybrid Automaton*) [8] - A hybrid automaton is a 7 tuple H = (*Loc, Var, f, Init, Inv, Jump,* ∑) whose components are:

1) A finite set *Loc* of discrete modes. It represents the discrete dynamics of *H*.
2) A finite set $Var = \{x_1, x_2, \ldots x_n\}$ of real valued continuous variables x_i, $1 \leq i \leq n$, $n \geq 0$. It represents the continuous dynamics.
3) 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 $\dot{x} = f(l, x)$, $l \in Loc$, $x \in Var$.
4) 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) consisting of a discrete mode $l \in Loc$ and a point $v \in \mathbb{R}^n$ being a valuation over *Var* which is a function $val: Var \to \mathbb{R}$.
5) The set $Inv \subseteq Loc \times \mathbb{R}^n$ called the invariant condition. As long as the *H* is in mode $l \in Loc$, the state must belong to *Inv*.
6) A set-valued function $Jump: Loc \times \mathbb{R}^n \to P(Loc \times \mathbb{R}^n)$ called the jump condition.
7) A finite set ∑ of events where each jump is represented by an event.

Definition *(Hybrid I/O automaton)* [9] - A hybrid I/O 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 $Var = Var_S \cup Var_I$.

Definition *(Linear and Affine Hybrid Automaton)* [10] – A *linear hybrid automaton* (LHA) is a hybrid automaton in which the invariants and the jumps are given by linear formulas over *Var* , and the flows are given by linear formulas over $\dot{Var}$. A *linear formula* is a finite disjunction of convex linear formulas and, a *convex linear formula* is a finite conjunction of constraints $\sum_i a_i x_i + b \bowtie 0$, with $a_i, b \in Z$, $x_i \in Var$ and $\bowtie \in \{\leq, <, =\}$ over the linear expressions $\sum_i a_i x_i + b$. Whereas, if the dynamics are given by *linear formulas* over the derivatives and the variables, then it is called as affine hybrid automaton (AHA).

Definition *(Rate Translation)* [5] - 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.

A. PHAVer

PHAVer (Polyhedral Hybrid Automaton Verifyer) [3] is a tool for 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 Hybrid I/O automata framework with affine dynamics. As PHAVer's computations are based on LHA, it 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. The abstractions like convex-hull used for simplification of polyhedra results in the forced termination of the algorithm in few cases. PHAVer also supports compositional reasoning.

B. HSolver

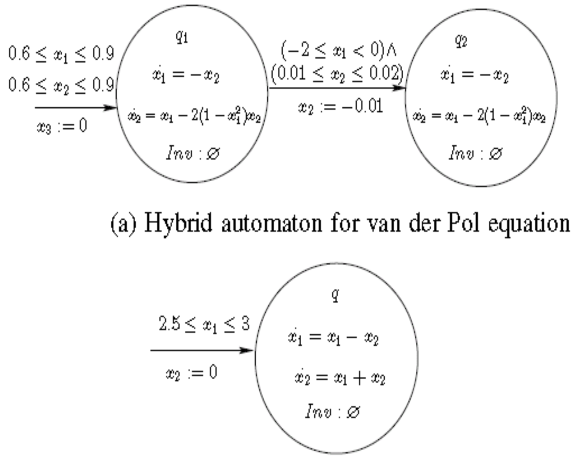
HSolver, a safety verification tool for nonlinear hybrid systems, was developed by Stephan et al. [4] based on a package RSolver [11] that provides pruning and solving of quantified constraints of the real numbers. The state space is divided into rectangular grids and, interval arithmetic is used to 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 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 [12]. Let θ denotes the angle the pendulum makes with the vertical, *l* the length of the pendulum, *m* its mass, and *K* the damping coefficient. The nonlinear system can be 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: $[-1.048, 1.048] \times [0, 1.2]$.



(a) Hybrid automaton for van der Pol equation

(b) Focus hybrid automaton

Fig. 1. Hybrid automata for van der Pol and focus benchmarks.

Linear Approximation: It is important to mention that $\sin(x_1)$ is almost equal to x_1 for very small values of x_1 i.e., $-1.048 \le x_1 \le 1.048$. However, to substitute $sin(\theta)$ by θ, we need to have a scaling factor z to compensate this linearization. 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. Assuming $\theta_l \le \theta \le \theta_u$, we can calculate the scaling factor $\theta/sin(\theta)$ over few values in this given interval and take an average of these. This mean serves as a scaling factor z for this location. Now, the % linearization error (δ) is given 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 [13]. The train moves on a circular track of length l. 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: $((s = 1 \vee s = 2 \vee s = 5) \rightarrow (5 \le \dot{y} \le 10 \wedge \dot{x} = \frac{1-x}{2})) \wedge ((s = 3 \vee s = 4) \rightarrow (5 \le \dot{y} \le 10 \wedge \dot{x} = \frac{10-x}{2}))$
- Jump: $((s = 1 \wedge y = 5) \rightarrow (s' = 2)) \vee ((s = 2 \wedge y = 15) \rightarrow (s' = 5)) \vee ((s = 2 \wedge y \le 15) \rightarrow (s' = 3)) \vee ((s = 3 \wedge y = 15) \rightarrow (s' = 4)) \vee ((s = 4 \wedge y = 5) \rightarrow (s' = 3)) \vee ((s = 4 \wedge y \le 5) \rightarrow (s' = 1)) \vee ((s = 5 \wedge y \le 5) \rightarrow (s' = 1))$
- 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, 3 dimensional example of a room heating problem defined by 3 rooms and 2 heaters [14]. The aim is to maintain a minimum specified temperature in each room. And, if the temperature in a room falls below a threshold, then either heater is turned on 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 it is switched on/off.

D. Van Der Pol Equation

We consider a 3-dimensional van der Pol second order equation with a time variable and some discrete jumps [4]. The hybrid automaton, shown in Fig 1(a), is explained as

- Flow: $(\dot{x}_1, \dot{x}_2) = (-x_2, x_1 - 2(1 - x_1^2)x_2)$
- Jump: $((s = 1 \wedge -2 \le x_1 < 0 \wedge 0.01 \le x_2 \le 0.02) \rightarrow (s' = 2 \wedge |x_2| \le 0.01 \wedge x_1' = x_1 \wedge x_3' = x_3))$
- Init: $0.6 \le x_1 \le 0.9 \wedge 0.6 \le x_2 \le 0.9 \wedge x_3 = 0$
- Unsafe: $(1 < x_1 \le 2) \wedge (0.01 \le x_2 \le 2)$
- 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 over approximated 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 over approximated by the dynamics of the form $x_1 - 2x_2 \le \dot{x}_2 \le x_1$ 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 [4] is shown in Fig. 1(b).

- Flow: $(\dot{x}_1, \dot{x}_2) = (x_1 - x_2, x_1 + x_2)$
- Empty Jump relation
- Init: $2.5 \le x_1 \le 3 \wedge x_2 = 0$
- Unsafe: $x_1 \le 2$
- State space: $[0, 4] \times [0, 4]$

F. Billiards Game

A classical example of a linear system consisting of a billiards table with a grey and a white ball [1]. Initially, the balls are placed at the positions $p_g = (x_g, y_g)$ and $p_w = (x_w, y_w)$ respectively. The grey ball is kicked and moves with a constant velocity. The ball rebounds as soon as it reaches the table boundary. We have defined the following unsafe condition: (1) whether this grey ball hits the white ball, 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: $x_g = 0 \wedge y_g = 0$

TABLE I: EXPERIMENTAL RESULTS ON PHAVER

PHAVer			
Benchmarks	Safety	Time(s)	Memory(KB)
Damping Pendulum	Unsafe Safe	0.10 0.25 (Refineme	1328 2784
Train-gate-controller	Safe	0.633	4256
Room Heating	Unsafe	26.23 (Convex-Hu ll)	116672
van der Pol	Safe	0.15	2112
Billiards Game	Unsafe	0.70	3808
Billiards Game	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	12 2	Safe	0.14	1920
Train-gate-controlle	2	167 5	Safe	1.50	3588
Room Heating [4]	1 4	248724	Unsafe	854.05	19720
van der Pol [4]	1	38	Safe	0.11	1212
Billiards Game 1	2 4	164718	Unknow n	77.06	2600
Billiards Game 2	1	22 3	Safe	0.20	2404
Focus [4]	8	94 2	Safe	0.40	1880

- 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 at 1.83 GHz and 2.0 GB of memory. PHAVer allows us to model only linear systems. Therefore, nonlinear systems have to be linearized to make their analysis possible using PHAVer. We have used the rate translation technique as in the damping pendulum and van der Pol equation 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 of our system because the linearization error has not been taken into account in most of the cases. Nevertheless, it helps us in comparing the functioning of two tools. We also computed the maximum linearization error in the damping pendulum benchmark as 0.2%. Computational results are given in Table 1. The damping pendulum system is declared as unsafe due to the over-approximation. However, we can still control this approximation to an extent by means of refinement which in turn limits the over-approximation and gives better results. We can see that the damping pendulum as well as billiards game are proved to be safe after the application of refining method because it improves the precision. PHAVer has the feature of compositional reasoning which makes it effective and useful in a situation where more than one automaton interact. The train-gate-controller is one of these types of examples. But HSolver does not provide the means to compose hybrid automata. Therefore, the user has to manually compose the autamata before using the tool.

In the room heating benchmark, it took almost a minute to check the safety property with default polyhedra abstraction. We tried to utilize PHAVer's abstraction functionalities such as convex-hull and bounding-box. When convex-hull abstraction (REACH STOP USE CONVEX HULL ITER = 20) was employed, it accelerated the termination and halved the verification time, which specifies the maximum number of iterations during which the convex-hull over approximation is used. Similarly, in van der pol benchmark, bounding-box abstraction (REACH USE BBOX ITER = 5) helped in the termination that specifies the frequency with which the bounding-box over-approximation is used. PHAVer also provides the choices in the search strategies such as breadth-first and depth-first search. In van der pol benchmark, we observed that the depth-first search took more time for verification than the breadth-first search. We have also noticed that limiting the number of bits and constraints speeds up the termination. For all the examples, we can see that memory consumption by PHAVer is huge due to its use of polyhedral representation for the states.

On the other hand, HSolver facilitates the verification of nonlinear hybrid systems. Operations such as $*$, sqr, cos, sin can easily be encoded in the tool. Experimental outcomes are tabularized in Table 2. We can see that, nonlinear dynamics are verified quite fast testifying to the tool's success in its use of pruning strategy. It is clear from the results that the pruning algorithm 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 because it also removes the points in the reach set. But in case of linear dynamics, it may or may not succeed which is clear from the billiards game benchmark. It also takes a lot of time as in the room heating benchmark. In the original paper [4], it was shown 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. After every refinement step, the tool prints the number of boxes used for representing the abstraction of original hybrid system. For e.g., in focus benchmark represents that after the fourth refinement step, the abstraction

```
*** 4 s: 5 initvol: 1. unsafevol: 2.70499080831 ***
```

consists of 5 boxes. In addition, initvol and unsafevol show the currently known upper bounds on the volume of all starting points and endpoints, respectively, of the error trajectories.

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 composition 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

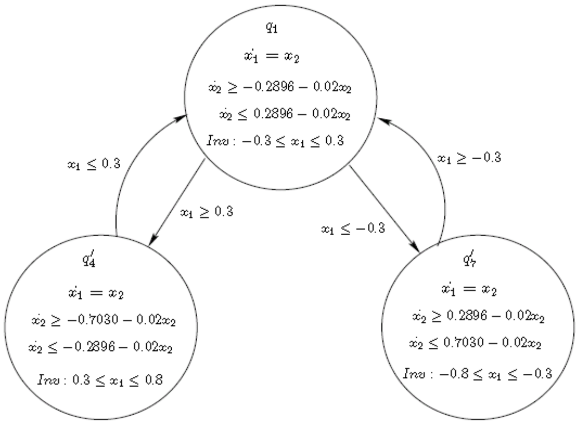


Fig. 2. An abstraction of the Pendulum benchmark

The refinement procedure could be held responsible. In nonlinear systems, we cannot 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 train-gate-controller example [10], we performed an existential quantification over the variables *x, t* and *y* using `reg = project_to(u)` where `reg` is the identifier 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*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 automata. We computed a simulation relation `rel = get_sim(`*H, H'*`)` between original pendulum example (*H*) and its abstraction (H') with fewer locations. The locations q_2, q_3 and q_4 in the original benchmark 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. Fig. 3) and so is the another abstract location q'_7. When tested, 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. [6] and Carloni et al. [7] 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 scenarios 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 state. 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.

Acknowledgment

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

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, no. 1, pp. 3–34, 6 February 1995.

[2] T. A. Henzinger, Pei-Hsin Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems, in *Orna Grumberg*, editor, CAV, vol. 1254 of *Lecture Notes in Computer Science*, pp. 460–463, Springer, 1997.

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

[4] S. Ratschan and Z. She, Safety verification of hybrid systems by constraint propagation based abstraction refinement, In M. Morari and L. Thiele, editors, HSCC, vol. 3414 of *Lecture Notes in Computer Science*, pp. 573–589. Springer, 2005.

[5] T. A. Henzinger and H. Wong-Toi, "Linear phaseportrait 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*, pp. 377–388. Springer, 1995.

[6] B. 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.

[7] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli, "Languages and tools for hybrid systems design," *Foundations and Trends in Electronic Design Automation,* vol. 1, no. 1/2, 2006.

[8] K. H. Johansson, *Hybrid Systems Lecture Notes.* UC Berkeley, Spring 2000.

[9] N. A. Lynch, R. Segala, and F. W. Vaandrager, "Hybrid I/O automata revisited," in Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, HSCC, vol. 2034 of *Lecture Notes in Computer Science*, pp. 403–417. Springer, 2001.

[10] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings of the Eleventh Annual IEEE Symposium On Logic In Computer Science* (LICS'96), New York, USA, 1996. *IEEE Computer Society Press*, pp. 278–293.

[11] S. Ratschan, "Efficient solving of quantified inequality constraints over the real numbers," *ACM Transaction Comput. Log*, vol. 7, no. 4, pp. 723–748, 2006.

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

[13] P. J. Antsaklis and Xenofon D. Koutsoukos, "Hybrid systems: Review and recent progress," in *Tariq Samad and Gary Balas*, editors, Software-Enabled Control, *Institute of Electrical and Electronics Engineer*, 2004, pp. 273–298.

[14] A. Fehnker and F. Ivancic, "Benchmarks for hybrid systems verification," in *Rajeev Alur and George J. Pappas*, editors, HSCC, vol. 2993 of *Lecture Notes in Computer Science*, pp. 326–341. Springer, 2004.

Manish Goyal received his M. Tech in Computer Science from Indian Institute of Technology Guwahati, India. His research interests include formal methods, model checking, logic and computation. Post his masters, he was associated with IBM India Labs, Bangalore as Associate Software Engineer. Thereafter, he worked at Verimag Research Lab, a leading research center in embedded systems in France. Currently, he is working at Atrenta Inc.