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 scenarios 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 system 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.
III. PRELIMINARIES

Definition (Hybrid Automaton) [8] - A hybrid automaton is a 7 tuple \( H = (Loc, Var, f, Init, Inv, Jump, \Sigma) \) 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 \( f: Loc \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) is called the vector field. It defines the continuous flow in each discrete mode \( l \in Loc \) through a differential equation \( \dot{x} = f(l, x) \).
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 \( val: Var \rightarrow \mathbb{R} \).
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 \( Jump: Loc \times \mathbb{R}^n \rightarrow P(Loc \times \mathbb{R}^n) \) called the jump condition.
7) A finite set \( \Sigma \) 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 \( Var_r \), and of output variables \( Var_o \subseteq Var_r \), where \( Var = Var_s \cup Var_r \).

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 \( Var \). A linear formula is a finite disjunction of convex linear formulas and, a convex linear formula is a finite conjunction of constraints \( \sum a_i x_i + b \leq 0 \), with \( a_i, b \in \mathbb{R} \), \( x_i \in \mathbb{R} \) and \( \mathbb{P} \subseteq \{\leq, =, \geq\} \) over the linear expressions \( \sum a_i x_i + b \). 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 Verifier) [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 \( \theta \) 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: \( x_1 = 1.048 \land x_2 = 1 \)
- Unsafe: \( x_2 \leq 0 \)
- State space: \([-1.048, 1.048] \times [0, 1.2] \).
Linear Approximation: It is important to mention that sin(θ) is almost equal to θ for very small values of θ, i.e., -1.048 ≤ θ ≤ 1.048. However, to substitute sin(θ) by θ, we need to have a scaling factor ω to compensate this linearization. We refine the rate over x1 into few intervals so that in each interval the scaling factor makes this linear approximation as close as possible to the original system. Assuming θ1 ≤ θ ≤ θn, we can calculate the scaling factor ω*sin(θ) over few values in this given interval and take an average of these. This mean serves as a scaling factor ω for this location. Now, the % linearization error (δ) is given as

\[ \delta = \left| \frac{\theta + \omega - \sin(\theta)}{\sin(\theta)} \right| \times 100 \]

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: \((s = 1 \land s = 2 \land s = 5) \rightarrow (5 \leq y \leq 10 \land x = \frac{1-x}{2}) \land (s = 3 \land s = 4) \rightarrow (5 \leq y \leq 10 \land x = \frac{10-y}{2})\)
- Jump: \((s = 1 \land y = 5) \rightarrow (s = 2) \lor (s = 2 \land y = 15) \rightarrow (s = 3) \lor (s = 3 \land y = 15) \rightarrow (s = 4) \lor (s = 4 \land y = 5) \rightarrow (s = 3) \lor (s = 4 \land y = 5) \rightarrow (s = 1) \lor (s = 5 \land y = 5) \rightarrow (s = 1)\)
- Init: x = 1, y = 0
- Unsafe: x < 5 \land y = 0
- State space: [0, 25] \land [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 with 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: \((s = 1 \land -2 \leq x_1 < 0 \land 0.01 \leq x_2 \leq 0.02) \rightarrow (s' = 2 \land |x_2| \leq 0.01 \land x_1 = x_1 \land x_1 = (x_1))\)
- Init: 0.6 ≤ x1 ≤ 0.9 \land 0.6 ≤ x2 ≤ 0.9 \land x3 = 0
- Unsafe: (1 < x1 ≤ 2) \land (0.01 ≤ x2 ≤ 2)
- State space: (1, -2, 2)[0.01, 2][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 x1. After the partitioning, the nonlinear dynamics are over approximated by linear flow. For example, the dynamics of the form \(x_{t+1} = x_t - 2(1-x_t^2)x_t\) in a location with state space \([0, 1] \times [0.01, 2] \times [0, 6]\) can be over approximated by the dynamics of the form \(x_{t+1} = 2x_t\) as x1 ≤ 3 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 ≤ x1 ≤ 3, x2 = 0
- Unsafe: x1 ≤ 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: \((q_1 \rightarrow (\dot{x} = 2 \land \dot{y} = 1)) \land (q_2 \rightarrow (\dot{x} = -2 \land \dot{y} = 1)) \land (q_3 \rightarrow (\dot{x} = 2 \land \dot{y} = -1)) \land (q_4 \rightarrow (\dot{x} = -2 \land \dot{y} = -1))\)
- Jump: The transition takes place as soon as the ball hits boundary of the table.
- Init: \(x_0 = 0 \land y_0 = 0\)
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 automata 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 +, -, *, 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. unsafenvol: 2.70499080831 ***

consists of 5 boxes. In addition, initvol and unsafenvol 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

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 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 $q_3, q_4$ and $q_6$ in the original benchmark are abstracted to $q_2, q_{4'}$ with the bound $0.3 \leq x_j \leq 0.8$ and the dynamics $(-0.7030 - 0.02x_3) \leq x_2 \leq (-0.2896 - 0.02x_2)$ 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 $q'$. 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.

![Fig. 2. An abstraction of the Pendulum benchmark](image)

### 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 Makhlof 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


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.

Manish Goyal