 doi:10.20944/preprints202103.0204.v1

A Review of Hybrid Automata Models

Mohammad Reza Besharati, Sharif University of Technology, Tehran, Iran, besharati@ce.sharif.edu

Dr. Mohammad Izadi, Assistant Professor, Sharif University of Technology, Tehran, Iran

Abstract

In this paper, Hybrid Automata, which is a formal model for hybrid systems, has been introduced. A summary of its theory is presented and some of its special and important classes are listed and some properties that can be studied and checked for it are mentioned. Finally, the purposes of use, the most widely used areas, and the tools that provide H.A. Support are addressed.

Keywords: Hybrid Automata, Formal Modeling, Discrete State, Continuous State, Formal Methods

1. Introduction

1.1. Definition of Hybrid Automation

In Automata theory, the Hybrid Automata is the official model for accurately describing composite systems that have both discrete computational and continuous processes. This model is a combination of a finite state machine and a set of differential equations [3] [2] [1].

Dynamic systems usually have both continuous behavioral and discrete behavioral aspects. Abstraction in discrete space should be used to model discrete aspects and abstraction in continuous space should be used to model continuous aspects. Many computer systems (such as embedded systems) and non-computer systems (such as biological systems) have this variety of behavior.

The modeling method of these systems should be aware of this diversity and be able to model discrete and continuous aspects simultaneously. Hybrid Automata models have this feature.

1.1.1. The first definition

A Hybrid Automation, named like H, consists of the following components [1]:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

Continuous state space variables: A finite set of variables in the form $X = \{x_1, \ldots, x_n\}$, each of which x_i is a variable of type real numbers. The number of these variables, ie n, is called the Hybrid Automata dimension. For continuous changes, we display the derivative of these variables as $X = \{x_1,\ldots, x_n\}$. For discrete changes, we display the values of these variables after the discrete change occurs with $X' = \{x_1',\ldots, x_n'\}$.

Discrete State Space and Its Transition Logic (Control Graph): A finite directional graph (V, E) in which V, the sum of the vertices of this graph, are states in discrete space for our system and in some sources are called Control states. Also, E, the edges of this graph, represent the transition between states in discrete space, and in some sources the edges of this graph are called Control Switches.

Initial conditions, invariant conditions and flow conditions: These conditions and states are labeled through three labeling functions on the outline of the graph, namely Control states. That is, the three labeling functions init, inv, and flow attach three statements to each discrete state (= each vertex of the graph).

For each vertex v, the proposition (init (v) is a proposition whose free variables are from the set X. This proposition represents the initial value of the variables of the continuous state space for this vertex (= this Control Mode). The vertex v is the initial state of discrete control (initial control mode), the initial values for the variables of the continuous state space are determined by the statement init (v), i.e. the ordered pair (v, init (v)) of a member of the set of states In some sources, the set of initial states of the system is called Init, so we can say: (v,init(v)).

For each vertex v, the proposition inv (v) is a proposition whose free variables are of set X. This statement states the necessary condition for survival in a discrete state (= staying in a Control Mode) based on the values of continuous space variables. If inv (v) is violated, the system cannot remain in the v discrete state. Of course inv (v) is only a necessary condition and not sufficient. That is, inv (v) may be established and the discrete state v may be left.

For each vertex v, flow (v) is a proposition whose free variables are from the set X Y X, and in fact this proposition is made up of a number of differential equations that are placed together in the form of a proposition. At the vertex v, the statement flow (v) represents Continuous Dynamics for continuous variables of the system, i.e. the logic of change in the values of continuous variables (which determine the continuous state of the system), in each discrete state v is determined by flow (v) The change in the continuous values of the continuous state variables is called "flow"

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

or "evolution of the state in continuous space", hence the name of the function that expresses the logic of changing the continuous variables for each discrete state. Of course, some sources use only the name Continuous Dynamics to refer to these propositions.

It should be noted that in a Hybrid Automata, continuous state variables are constantly changing. In any discrete state (= or Control Mode), continuous variables change in a way and with a specific logic of that state. How to change continuous variables in any discrete case v is expressed through differential equations and in the form of flow (v). Hence, some sources have stated that each discrete state of a Hybrid Automata is assigned a set of differential equations that express the Continuous Dynamics of the system for that state. In other words, the system consists of a number of states (= Statics) and the change of these states (= (Dynamics). Each HA actually represents a statics and its Dynamics. Therefore, HA can be considered an independent model from this perspective. Know the concept of time, because Dynamics is only an order on the statics space (= represents the order of change of states). If we do, then we can consider HA as a temporal model, so depending on how we look at it, HA can be considered both a temporal model and a time-independent model, which is a very interesting flexibility Some of them use it only as a logical and high-level modeling to express the logic of Statics and Dynamics of the system, and some even use it as a competitor to Timed Automata - that is, very pragmatic and close. Take advantage of the time concerns of implementing a system.

Jump Conditions: These conditions are defined by a tagging function called jump. This function attaches a statement to each edge of the control graph (= to each Control Switch). For edge e, the free variables in the jump statement (e) are from the set X Y X '.

Some sources have divided the jump conditions into two preconditions, precondition and post-condition. Formally, however, there is no need to distinguish between preconditions and post-conditions by using X variables alongside X variables. Because the preconditions are assigned to the members of X 'and the preconditions are around the members of X'. When we can use the terms X Y X 'in condition statements, it means that we can express the meanings of precondition and post-condition together in the form of a condition.

These conditions are in fact conditions that are necessary for a discrete change to occur. Here, too, these conditions are only necessary and not sufficient. That is, there may be conditions for a discrete change, but that discrete change does not occur. To

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

model a necessary and sufficient condition for a discrete variable, one must use the coordinated setting of the conditions for survival in a state (previously introduced in inv (v)) and the jump conditions of the output edges of that state. As we said before, inv (v) expresses the conditions for survival in a discrete state. Therefore, if we reach a state where inv (v) is not present, we have to leave the v state. Therefore, we must select one of the output edges of v that has the jump conditions of that edge. Now, in this situation, if only the jump condition of one of the edges is met, we have to cross that edge. So by simultaneously adjusting inv and jump, the necessary and sufficient conditions for the occurrence of a discrete change in HA can be modeled and included.

Events: In each HA we have a finite set of events called Σ. An edge tagging function called event: E ϕ Σ assigns an event to each edge in the control graph (= to each Control Switch, i.e. to each transition in the discrete state of the system). Events can be viewed through the discrete inputs of the system. In a way, in some sources, instead of using the word event, the word "discrete inputs" is used to describe the set Σ in the definition of HA.

1.1.2. The second definition

Hybrid Automata is an 8-tuple as follows [4]:

$$H = (\mathcal{Q}, \mathcal{X}, \boldsymbol{f}, Init, Inv, \mathcal{E}, \mathcal{G}, \mathcal{R})$$

- Q = $\{q_1,\ldots,q_n\}$ is a finite set of discrete states.
- X is a continuous state space. A space P^n where n is the dimension of Hybrid Automata.
- A vector field that, for each discrete state, expresses the transition logic of continuous states:

  $$\boldsymbol{f} : \mathcal{Q} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$$

- Set of initial system states:

  $$Init \subset \mathcal{Q} \times \mathbb{R}^n$$

- Invariant conditions, i.e. the conditions necessary for continuous variables to survive in discrete states

  $$Inv : \mathcal{Q} \rightarrow 2^{\mathbb{R}^n}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

- A transition relation that expresses the logic of transition between discrete states:

  $$\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$$

- Guard conditions on transitions:

  $$\mathcal{G} : \mathcal{E} \rightarrow 2^{\mathbb{R}^n}$$

Reset Map: During discrete transitions, the values of the continuous variables are reset according to this map. (In some sources, instead of introducing this map, transition conditions are used to reset the values during the transition. But in this definition, this concern is separated from the transition condition concern and is listed in a separate component called the reset map.)

$$\mathcal{R} : \mathcal{E} \rightarrow 2^{\mathbb{R}^n} \times 2^{\mathbb{R}^n}$$

Some sources add two more components to the aforementioned eight components, giving a Hybrid Automata a dozen. These two components are:

Set Σ (referred to by different names in different sources: a set of events, the alphabet of discrete inputs, or synchronization labels. All of these have a single meaning and use the Σ symbol to represent it.)

The λ tag function that attaches a member of Σ to each state transition in the discrete state space (= each edge of the control graph). This labeling function is useful for defining the simultaneous execution of two Hybrid Automata and their Composition.

1.1.3. Third definition

The dynamics of a Hybrid System can be modeled with a model called Hybrid Automata, which consists of the following entities [5]:

- Discrete State Space: q ε Q where Q is a finite set.
- Continuous State Space: x ε P^n
- Discrete input: Σ members
- Continuous input: v ε V which is V ζ P^n.
- Continuous Dynamics:
  $$\dot{x} = f(q, x, v)$$
- Discrete dynamics:

  $$R : Q \times \mathbb{R}^n \times \Sigma \times V \rightarrow 2^{Q \times X}$$

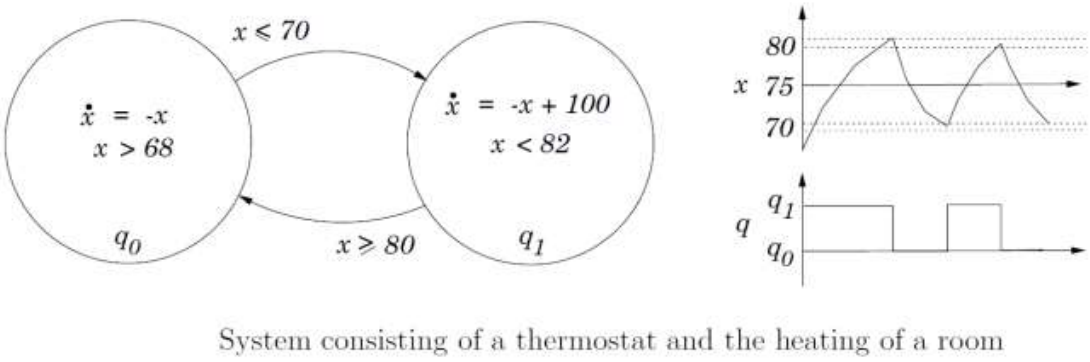
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

- Invariant conditions: These conditions actually express the meanings of any discrete state in the form of conditions necessary for survival in that state. These meanings are based on a combination of states and inputs as follows:

$$Inv \subseteq Q \times \mathbb{R}^n \times \Sigma \times V$$

- Initial cases:

$$Init \subseteq Q \times \mathbb{R}^n$$



System consisting of a thermostat and the heating of a room

Figure 1. An example of a H.A. model that models a thermostat system. The variable x represents the temperature of the thermostat. On the right, the state transition diagram is plotted over time for an example system execution for continuous and discrete states [5].

1.2. The various components of a Hybrid System in Hybrid Automata

Each system can be composed of a number of components, the dynamics of these components and the relationship between them. A Hybrid System consists of two types of components - continuous and discrete. In other words, any system can be viewed in the form of statics, the dynamics of these components and the relationship between the two.

In a Hybrid Automata, discrete states represent the discrete components of a Hybrid System. Continuous state variables also represent its continuous components. Together, these two components model the statics of a system.

In a Hybrid Automata, the logic of transition between discrete states (= edges in the control graph or Control Switches) indicates how dynamic the discrete components of the system (Discrete Dynamics) are. Differential equations that determine and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

describe how the system variables change for each discrete state also express how the continuous components of the system (Continuous Dynamics) [5].

The relationship between discrete components and continuous components and the relationship between the dynamics of the two is explained by jumping conditions on the edges, invariant conditions and how the flow state is assigned to discrete states (= differential equations written on each vertex of the control graph).

Another concern of system modeling is the issue of system inputs and outputs, which is predicted in Hybrid Automata modeling of discrete and continuous inputs - both. The output problem can also be modeled in the form of system state variables.

Thus, from the perspective of the first level of system vision, the Hybrid Automata can be considered a complete system modeling that addresses both the continuity and disintegration concerns of system components, dynamics, and communications.

1.3. Types of Semantics

Hybrid Automata theory supports Safe Semantics and Live Semantics [1].

To define Safe Semantics and Live Semantics, the Transition System definition is used on the state space (on the state space set, i.e. on the Cartesian multiplication of the continuous state space and on the set of discrete state values). Of course, for each of these two semantics, the Transition System is defined slightly differently [1].

From a chronological point of view, two different views can be given to the meanings of a H.A. Had. From one point of view, you can have a very timely and so-called Timed Semantics view. That is, in the Transition System, which means H.A. Defines, for each transition in the state space, in addition to determining the logical condition of the transition, also specified the time period required for the occurrence of this transition [1]. That is, with such a view of the meaning of H.A., even a H.A. Considered a competitor to Timed Automata.

From another perspective, one can see Semantics as "abstracted from time" and "set aside time", the so-called Time-Abstract Semantics. That is, in the Transition System, which means H.A. Defines, for each transition in state space, we specify only the logical condition of the transition. Although we know there is a limited time frame in which this transition occurs, we do not know its magnitude. We only know that this transition is possible in terms of time. In fact, we only know the chronology between the scenarios and we do not know the details of the time and the length of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

time required for the transition to occur. In other words, we are fully aware of the phases of change but not the duration of the change [1].

2. Several specific Hybrid Automata classes

2.1. Zeno and NonZeno

In a H.A. And in its state space, we have both discrete and continuous transitions. Discrete transitions occur by passing over the edges of a control graph, and continuous transitions occur by continuously affecting the continuous variables of the state space. In H.A., there are continuous transitions that are tied to the concept of time. That is, depending on the way we look at the meaning of the system, either due to the passage of time, they occur continuously or time is defined based on their occurrence. Therefore, the concept of continuous time and transitions in H.A. Tied together. With this introduction, we come to the definition of Zeno behavior.

Zeno behavior is a phenomenon for a dynamic system that is defined as follows [4]: Infinite occurrence of events in a limited period of time[1]

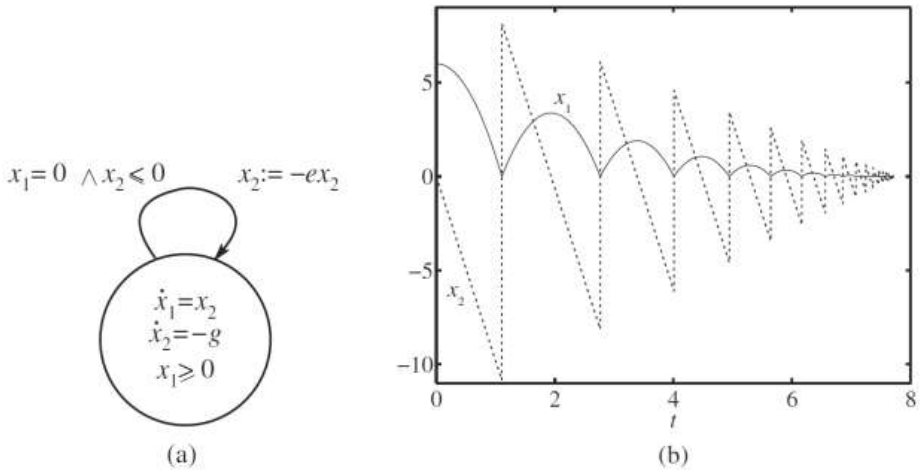
If H.A. performs infinitely discrete transitions over a limited period of time, it has Zeno behavior. In other words, if the infinitely discrete transition is associated with only a "finite magnitude"[2] of the transition in the space of continuous variables, the Zeno behavior of H.A. have seen.

Take this hybrid system as an example:

[1] This behavior was apparently first introduced by the famous Greek philosopher Xenon Elias in his famous paradox. Various forms of this paradox are presented. The paradox is this:
"For example, if Achilles, the most extremist of the people, runs after a turtle, he will never reach it. Because whenever Achilles walks a distance, the turtle walks the same distance, and Achilles has to walk that distance."
Or: "Achilles must first cover half the distance. This is the other half of the distance. Then he must cover the remaining half of the distance. Then he must first cover half of the remaining distance, which causes "Let the other half remain. This means that Achilles must do an infinite journey, which requires a non-zero time. Therefore, Achilles should never reach the turtle."

[2] In the independent view of time, time can be defined as "the magnitude of motion", that is, time is not original and is only the result of measuring and appreciating motion. It should be noted that Time-Abstract Semantics supports this view.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1



Bouncing ball: hybrid automaton (a) and simulation (b) with $x = x_1$

Figure 2. Example of a Zeno system [4]

A ball that lands on the ground and loses a constant fraction of its kinetic energy each time. This species must hit the ground infinitely to lose all its kinetic energy. A model H.A. Provided for this system. In this model, no assumptions are made about time. Therefore, if we attribute this model to our external intuition of the ball falling (= we choose this meaning and construction of meaning for this model), there will be no inconsistency in the construction and meaning of the model. But according to our intuition (the same semantic device attributed to the model that is not connected to the syntax structure of the model in terms of time), the ball will stand for a short time. According to the model, we will hit the ground infinitely before the ball stops. Hence, this model, with the semantic device chosen for it, creates an overall device with Zeno behavior.[3]

To H.A. Those in which Zeno behavior is not observed are called NonZeno. Only H.A. NonZeno devices can be implemented and built externally, hence H.A. NonZeno is a favorite of engineers.

[3] According to the author of these lines, if the Single Source of Truth principle is used alongside the Constructive approach to build a system (both the structure and the meaning of the system), it does not make sense for the Zeno paradox to emerge. Because with the Zeno paradox, by moving from the Single Source of truth and constructing different states (Constructive approach means transition of states), we must reach the limit on the one hand and Infinity on the other. By induction on the size of this constraint, it can be shown that in a construction operation from a single origin, it is not possible to reach the constraint on the one hand and the infinity on the other. From a Constructive perspective and in the Single Source of Truth paradigm, a system with Zeno behavior is not a Well Defined and Well Structured system and suffers from dualism and paradox.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

2.2. Stable and Instable

If a H.A. Like H on a region O of the state space, be stable. In this case, for each course of states (= a sequential process of continuous and discrete transitions[4]) such as r, there will be a time t0 after which the whole course The states will be within region O [4]:

Ar Et_0 At t_0 : r(t) ε O

Stability and instability of a H.A. Defined on an area.

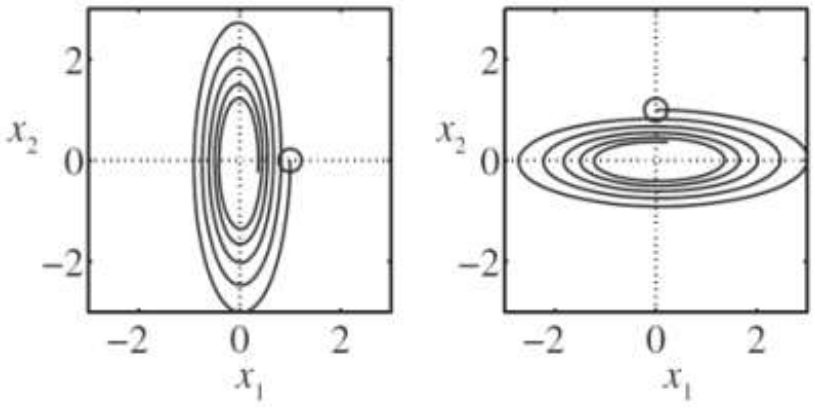


Figure 3. An example of a typical state course in the state space that shows the stability of the system in this area [4].

[4] a trajectory of a hybrid automaton is a succession of continuous and discrete transitions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

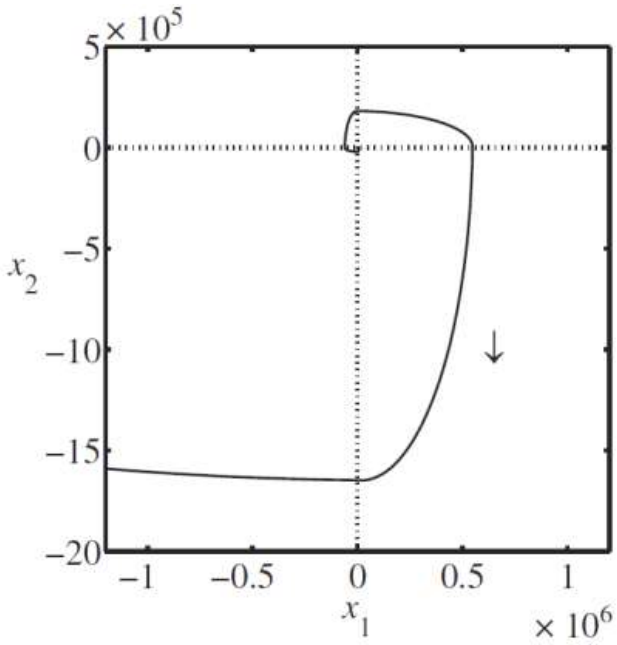


Figure 4. An example of a typical state course in the state space that shows the instability of the system [4].

2.3. Linear Hybrid Automata

A H.A. Which has the following two conditions is considered linear [1]:

1. The initial condition, the invariant condition, the flow condition and the jump condition of the Boolean combination of linear inequalities.

2. Differential equations in flow conditions (Flow Conditions) consist of only X and X. (Only have a derivative in them once.)

With the tools of HyTech and PHAVer, the following can be calculated and obtained for Linear H.A.s with accurate multidimensional[5] calculations [4]:

1. Set of Available States: Of course, in the general case, the accessibility issue for Linear H.A. It is indecisive, but in most practical examples, this is the case for Linear H.A. It is solvable. Even so, owning one is still beyond the reach of the average person.

[5] Polyhedral Computations

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

2. Synthesis of controllers

3. Check the Equivalence and Abstraction relations between Various Linear H.A.

Another approach to solving Linear H.A. There is use to the fact that the terms of these models are linear. Therefore, these models can be solved with linear programming techniques. Linear H.A. Solve with linear programming techniques and tools. An example is the ARMC tool. "[4]

2.4. Polyhedral-Invariant Hybrid Automata

A H.A. Which has the following three conditions is considered PIHA [6]:

1. Differential equations in flow conditions (Flow Conditions) are of ODE[6] type.

2. Any guard condition (Guard Condition) is a linear inequality.

3. The values of continuous variables do not change in discrete transitions. In other words, the reset conditions are all the same.

4. Inversion conditions and jump conditions are set in such a way that all jump conditions are False to survive in a control state (= a discrete state). That is, it cannot remain in a state if one of the jump conditions is True. The name of this group is H.A. It is taken from this condition.

[6] ordinary differential equation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

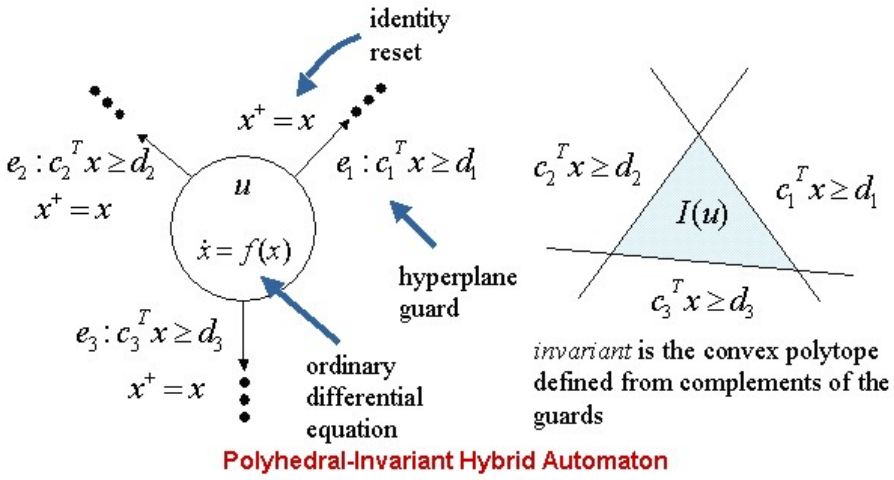


Figure 5. Characteristics of a PIHA [6]

CheckMate Verification Tool in MATLAB environment allows us to check PIHA models [6].

2.5. Rectangular Hybrid Automata

A subspace R of P^n is called a rectangle if:

$$R \subset \mathbb{R}^n \text{ , } R = \prod_{i=1}^{n} R_i$$

Ri are finite intervals whose two points at the beginning and end are rational numbers [7]. In summary, an R.H.A. has two general properties:

- Its subspaces are rectangular.
- Differential equations that describe the changes of continuous variables are independent of the discrete state. Also, the variables of the continuous state space are independent of each other.

Of course the exact definition of R.H.A. It has more details. But the above two properties make the issue of accessibility for R.H.A. The initialized values can be determined [8]. Hence R.H.A. A very important class of H.A. Are.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

3. Some verifiable properties for Hybrid Automata

Similar to other Automata [11], there are Liveness and Safety features for Hybrid Automata. Because Hybrid Automata theory supports Safe Semantics and Live Semantics, it is possible to check these properties for H.A. [1].

Only H.A. NonZeno devices can be implemented [1] and have engineering value. Using Live Semantics can be a Zeno or NonZeno of a H.A. Checked [1].

An important loop of the various analyzes on H.A. is the accessibility analysis on scenarios. Although accessibility has been shown to be an indecisive problem for the Hybrid Automata [8] [1], there are well-accepted and approximate algorithms, but non-coverage[7], for calculating the available areas of the state space [4].

An example of these algorithms is below.

Algorithm 3.1 Reachable space computation

Given: the initial region $(Init)$

Init: $\text{Reach}(Init) \Leftarrow Init, \mathcal{R} \Leftarrow \text{Succ}_C(Init)$

1: **while** $\mathcal{R} \not\subset \text{Reach}(Init)$ **do**
2: $\quad \mathcal{R}^* \Leftarrow New\,(\mathcal{R}, \text{Reach}(Init))$
3: $\quad \text{Reach}(Init) \Leftarrow \text{Reach}(Init) \cup \mathcal{R}$
4: $\quad \mathcal{R} \Leftarrow \text{Succ}_H(\mathcal{R}^*)$
5: **end while**

Result: $\text{Reach}(Init)$

Figure 6. An example of accessibility algorithms [4]

Another important loop of analysis on Hybrid Automata is the check and review of Trace Inclusion in a set of Traces of an Automata [1]. There are two possible modes. One is Timed Trace Inclusion and the other is Time-Abstract trace inclusion, the first of which is defined on Timed Semantics and the second on Time-Abstract Semantics [1].

Unfortunately, the issue of Timed Trace Inclusion is indecisive for any Timed Automata [1]. Hence the first way for H.A. It can also be shown that it is indecisive.

But the Time-Abstract Trace Inclusion problem for some specific classes of H.A. Are decisive [1].

[7] That is, they cannot find all the available states.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

Checking the Stability and Instability of H.A. It is another case to be considered [4].

Another issue, Composition of two or more H.A. Is. For example, in Mr. Henzinger [1], the meanings of Timed Semantics and Time-Abstract Semantics for Parallel Composition of two H.A. Provided. Another issue that can be checked for Hybrid Automata is the compatibility check. When several Hybrid Automata are merged, the compatibility of the automata resulting from the merger should be checked [1].

Simulation, Bi-simulation and Equivalence Relationships for H.A. In the theory of H.A. Has been defined [1] and one of the important cases that can be examined is the same relationships in the H.A. Is.

4. Purposes of using Hybrid Automata

Models of H.A. are official models for hybrid systems. Therefore, all the purposes of using formal modeling can be enumerated for them in the field of hybrid systems. Items such as [9] [4] [1]:

- Inspection and validation of hybrid systems and designs (such as Cyber-Physical Systems[10])
- Synthesis of controller and controller
- Optimal Control Process Optimal control over a specific objective function
- Hierarchical Control: means to divide and distribute the control process in a hierarchical structure.
- Multi-agent and distributed control
- checking the safety and liveness features
- Provide an integral computing unit for describing, analyzing and simulating hybrid systems
- Separating the logic concerns of hybrid subsystems from the rest of the concerns of a system

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2021 doi:10.20944/preprints202103.0204.v1

5. The most used areas of H.A.

Wherever there is a hybrid system, H.A. Used and have used. Items such as:

- Modeling of biological and physical systems
- Modeling of human, urban and traffic systems
- Modeling of complex systems
- Computer Graphics: A logical unit for the automatic description and synthesis of computer graphics governing physical rules

6. H.A. Tools

Among the tools used to describe, inspect, and even synthesize H.A. models. Supports can be [4] [6]:

- HyTech is one of the oldest and oldest tools in this field.
- Checkmate Verification Tool in MATLAB environment
- PHAVer

And linear programming tools that can be used to solve Linear H.A. models. used.

7. Conclusion

In this paper, Hybrid Automata was introduced, which is a formal model for hybrid systems. A summary of its theory was presented. Listed some of its special and important classes and pointed out some properties that can be studied and checked for it. Finally, for the purposes of use, the most widely used areas, and the tools provided by H.A. Support paid.

8. References

[1] Thomas A. Henzinger, **"The theory of hybrid automata"**, Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, pp. 278-292, 1996. An extended version appeared in Verification of Digital and Hybrid Systems (M.K. Inan, R.P. Kurshan, eds.), ASI Series F: Computer and Systems Sciences, Vol. 170, Springer, pp. 265-292, 2000.

[2] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho, **"Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems"**, In *Hybrid Systems I*, Lecture Notes in Computer Science 736, Springer, pp. 209-229, 1993.

[3] Christos G. Cassandras, John Lygeros, ***Stochastic hybrid systems***, Boca Raton : CRC/Taylor & Francis, 2007.

[4] Jan Lunze, Françoise Lamnabhi-Lagarrigue, ***Handbook Of Hybrid Systems Control***, Cambridge: Cambridge University Press, 2009.

[5] Claire J. Tomlin, "**AA278A Hybrid Systems: Modeling, Analysis, and Control Course, Lecture1: introduction"**, 2005, http://www.stanford.edu/class/aa278a/lecture1.pdf

[6] ***CheckMate Verification Tool Guideline***, Mathworks, http://www.mathworks.com/matlabcentral/fx_files/15441/3/content/doc/verify/content.htm

[7] Claire J. Tomlin, "**AA278A Hybrid Systems: Modeling, Analysis, and Control Course, Lecture5: Sequence Properties and Verification"**, 2005, http://www.stanford.edu/class/aa278a/lecture5.pdf

[8] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya, **"What's decidable about hybrid automata?",** *Journal of Computer and System Sciences* 57:94-124, 1998.

[9] Claire J. Tomlin, John Lygeros, and S. Shankar Sastry, **"A game theoretic approach to controller design for hybrid systems"**, *Proceedings of the IEEE* 88, No.7,pp. 949-970, 2000.

[10] Ardeshir-Larijani, Ebrahim, Alireza Farhadi, and Farhad Arbab. "**Simulation of Hybrid Reo Connectors**." In *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1-10. IEEE, 2020.

[11] Bonsangue, Marcello M., and Mohammad Izadi. "**Automata based model checking for Reo connectors**." In *International Conference on Fundamentals of Software Engineering*, pp. 260-275. Springer, Berlin, Heidelberg, 2009.