Hybrid Automata in Twin Activate

Hybrid systems define a very wide class of dynamical systems that involve the interaction of heterogeneous data types and dynamics especially the interaction of continuous-time dynamics described by differential equations, with discrete dynamics described by a finite state under automata or other models of computation. Examples include mechanical systems with collisions, circuits with ideal diodes and switches, chemical processes controlled by valves or pumps, and most importantly, embedded computation systems where digital devices interact with an analog environment.

Early works on formal models for hybrid systems includes phase transition systems and hybrid automata. A hybrid automaton is a state machine augmented with differential equations. It is a standard model for describing a hybrid system. Hybrid automata come in several forms: The Alur-Henzinger hybrid automaton is a popular model that was developed primarily for algorithmic analysis of hybrid systems model checking [1, 2, 3]. In this document we will explain the way **Twin Activate** environment can be used to model and simulate hybrid systems, and in particular, hybrid automata.

Hybrid automata have been introduced by Xavier Nicollin *et al*. in [5] and an analysis of linear hybrid automata are given in [2, 4]. They modeled a hybrid system as a finite automaton that is expanded with a set of real valued variables. These variables can be tested and modified at transitions. At an automaton mode the value of variables change continuously with time according to evolution laws which are associated with the mode. The transition relations are specified by guarded commands; the activities by differential equations; and the invariants by logical formulas. We now present the relevant definition of a hybrid automaton from Henzinger's theory [4] that will be used in the construction of our formal semantics for **Twin Activate** automaton block.

H is a tuple which is defined as:

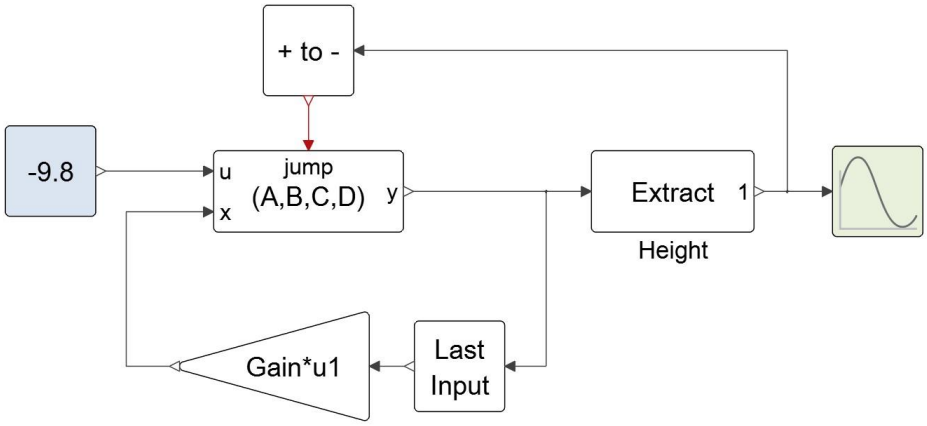
H = $(V,E,X,F,Invariant,Initial,Jump)$, where

- V is the set of control modes $\{v_1,\ldots,v_M\}$ where M is the number of control modes;
- E is the set of control switches which identify the source mode and a destination mode during a transition.
- X is a set of real-valued variables $\{x_1,\ldots,x_N\}$ where N is the dimension of H. $\dot{X}$ is first time derivative of X;
- F is a predicate over $\{X,\dot{X}\}$ assigned to each control mode; $Invariant$ defines the admissible range for $\{X,\dot{X}\}$ in each mode;
- $Initial$ defines initial values of $\{X,\dot{X}\}$ in each mode.
- $Jump$ is an edge labeling function which assigns a predicate over $\{X,\dot{X}\}$ to each edge.

As an example consider the model of the bouncing ball

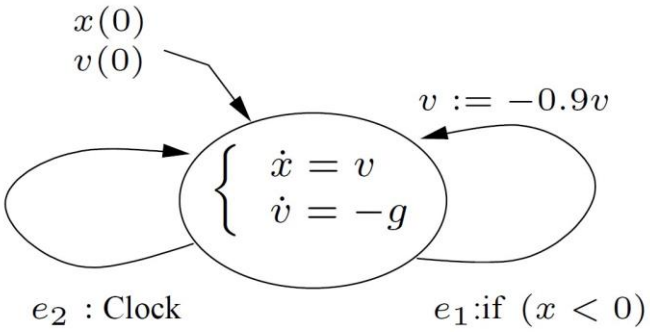
$$\begin{cases} \dot{x} &= v \\ \dot{v} &= -g \end{cases}$$

The model of this bouncing ball has been implemented in **Twin Activate** looks as follows



The output of the block JumpStateSpace is a vector of size two consisting of $[x,v]^t$. Whenever the condition ($x < 0$) is satisfied, an event is generated by the Zero-crossing (**+** to **-**) block. This event makes the Jump block do a reinitialization, *i. e.,* $v := -0.9v$ and $x := x$. The Clock block generates periodic events (activation signals) to activate the Scope block which displays x and v.

This hybrid system can also be modeled as an automaton with a single control mode. The graphical representation of this automaton is depicted in



There are two variables x and v whose initial values are given at $t = 0$. There are also two event sources; a zero-crossing event and a periodic time event. Only the zero-crossing event causes discontinuity in variables.

Although it is often possible to model hybrid automata with generic **Twin Activate** blocks, it is not an easy and efficient way to develop a hybrid automaton especially when the number of control modes is high. In a hybrid automaton composed of several control modes, at any time instant, only one mode or subsystem is active and the others should stay inactive. In **Twin Activate**, even though it is possible to generate an activation signal to activate conditionally a subsystem via If-Then-Else blocks, the variables are still present in the state vector of the model and reduce the performance of the numerical solver. Another difficulty is in detecting the jumps; in each control mode of the hybrid automaton, zero-crossing functions should be evaluated for monitoring the jump conditions. If we develop a hybrid automaton with several subsystems activated by If-Then-Else blocks, all zero-crossing functions are active and it will be difficult to distinguish the zero-crossing functions that have to be monitored. Using switches and selector blocks, to activate or deactivate the zero-crossings, increases the model size and reduces the simulation speed.

Twin Activate is a hybrid system simulator which means that **Twin Activate** can simulate mixed continuous-time and discrete-time subsystems. Developing hybrid automata containing large number of modes and continuous-

time states [using generic **Twin Activate** blocks] is a cumbersome, timeconsuming, and difficult task. The Automaton block provides a modular way to model a general hybrid automaton in **Twin Activate**. This block provides an interface to model and encode the graphical representation of an automaton to be used by the simulator and consequently by the numerical solver in a completely transparent way.

Bibliography

[1] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Angiovanni-Vincentelli, "Languages and tools for hybrid systems design,"Theoretical Computer Science, vol. 1, pp. 1-193, 2006.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in Hybrid Systems, 1992, pp. 209-229. [Online]. Available: citeseer.ist.psu.edu/alur92hybrid.html

[3] N. Lynch, R. Segala, F. Vaandrager, and H. Weinberg, "Languages and tools for hybrid systems design in hybrid systems iii," Lecture Notes in Computer Science„ vol. 1066, pp. 496-510, 1996.

[4] T. A. Henzinger, "The theory of hybrid automata," Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), pp. 278-292, 1996.

[5] X. Nicollin, J. Sifakis, and S. Yovine, "From ATP to timed graphs and hybrid systems," Acta Informatica, vol. 30, no. 2, pp. 181-202, 1993. [Online]. Available: citeseer.ist.psu.edu/article/nicollin93from.html

[6] K. J. Astrom and K. Furuta, "Swinging up a pendulum by energy control," Automatica, vol. 36, pp. 278-285, 2000.

[7] M. di Bernardo, F. Garofalo, L. Glielmo, and F. Vasca, "Switching, bifurcations, and chaos in DC/DC converters," IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, vol. 45, no. 2, pp. 133-141, Feb. 1998.

[8] E. Fossas and G. Olivar, "Study of chaos in the buck converter," IEEE Trans. on Circuits and Systems I, vol. 43, pp. 13-25, 1996.

[9] F. E. guezar, ascal Acco, H. Bouzahir, and D. Fournier-Prunaret, "Chaotic behavior in a hybrid dynamical system that arises from electronics," AIMS' Sixth International Conference on Dyn. Systems, Diff. Equations and Applications, Poitiers, France, 2006. [10] http://sec.eecs.berkeley.edu/demo/StickyBall/StickyBall.htm