Some time leaks in hybrid modelers (short talk)

Marc Pouzet
DI, École normale supérieure
Marc.Pouzet@ens.fr

Workshop SYNCHRON
Le Croisic, 20 nov. 2012

Programming with mixed signals

Program both discrete and continuous-time systems. E.g:

- a discrete/continuous controller and a discrete/continuous environment;
- a continuous environment with several modes (clutch-control, etc.).

Several tools already exist and are widely used:

- Simulink/Stateflow, LabVIEW, Modelica, etc.
- VHDL-AMS, VERILOG-AMS, etc.

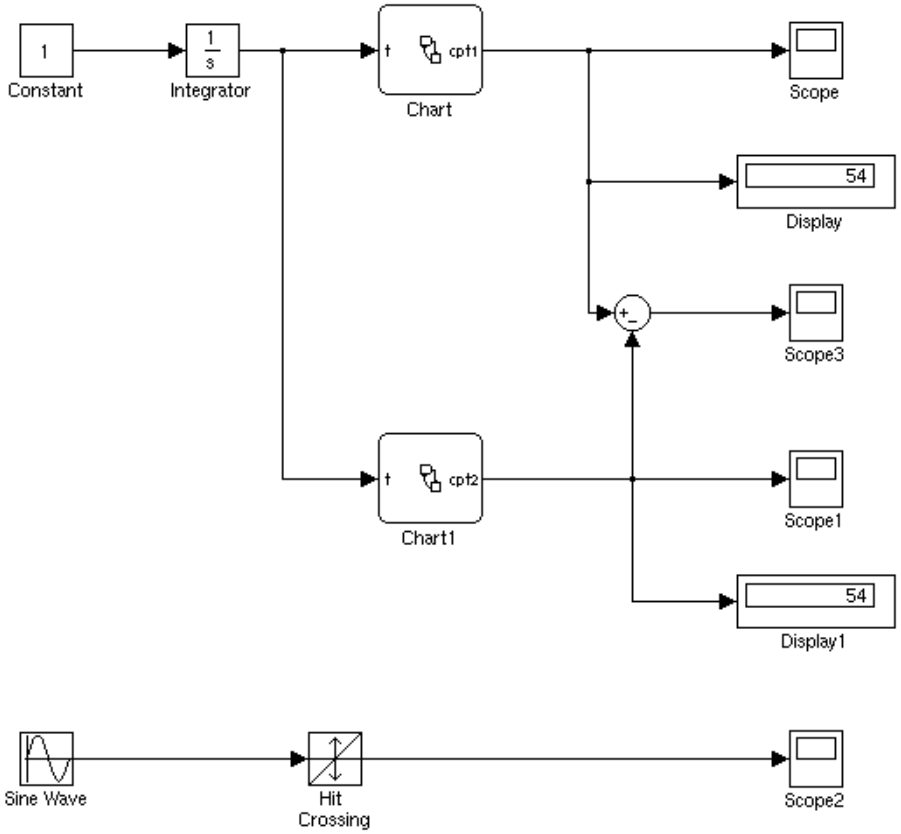
But: No comprehensive semantics. This is a hard problem as a numeric variable-step solver and floatting-point numbers come into play. A semantics exists for discrete subsets only (e.g., [Caspi et al., Hamon and Rushby, Hamon] for Simulink/Stateflow).

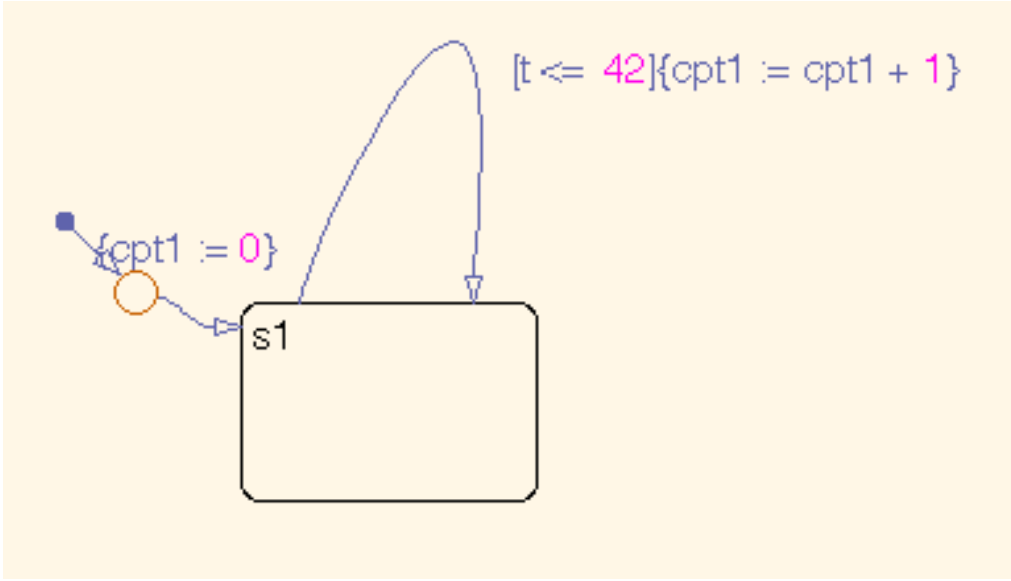
Mosterman et al. made several important clarifications on the behaviour of hybrid modelers continuous/discrete interactions.

Time leaks between continuous and discrete: One problem is that discrete time is not logical but global: it exposes the internal steps of the simulation engine. This causes strange behaviours.

Strange behaviours of Hybrid modelers

E.g., Simulink/Stateflow. What is a discrete step? How long is it?

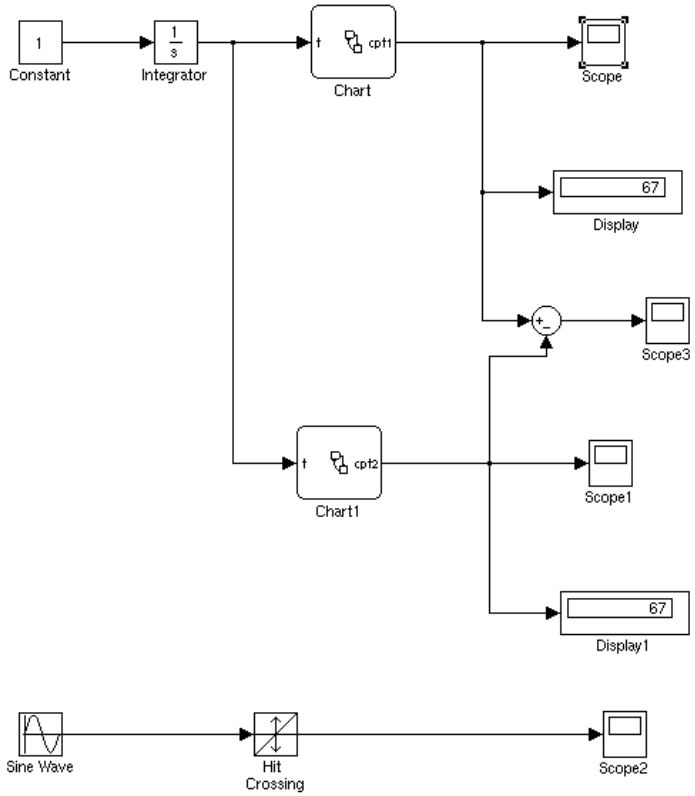


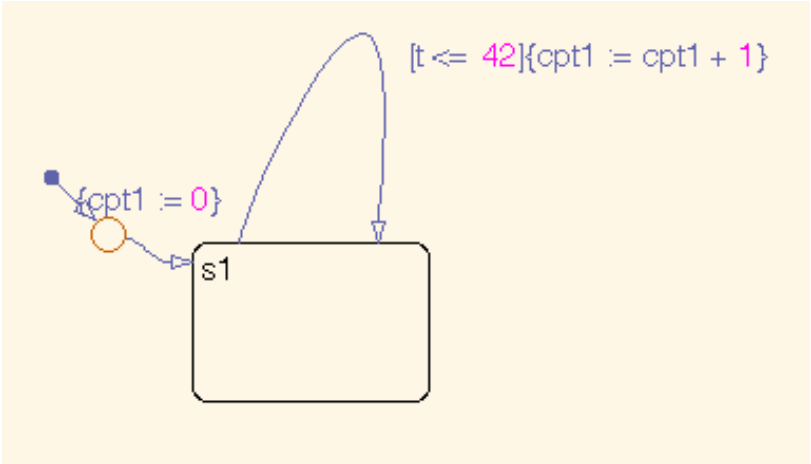


Select solver `ode23s` (stiff/Mod. Rosenbrock); Amp = 1; Freq = 1 (sinusoid).

Strange behaviours of Hybrid modelers

E.g., Simulink/Stateflow.

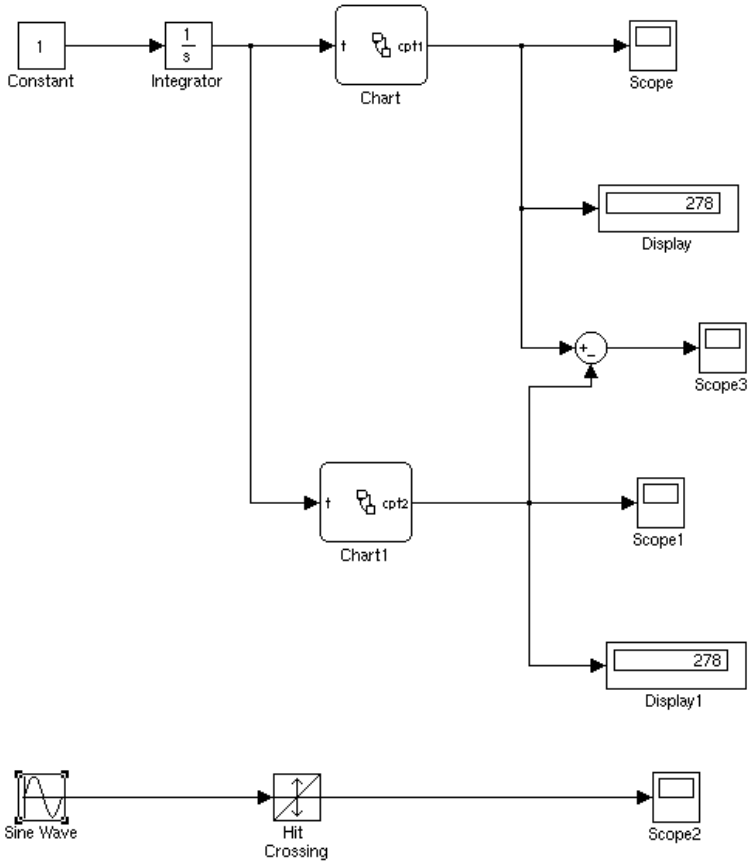


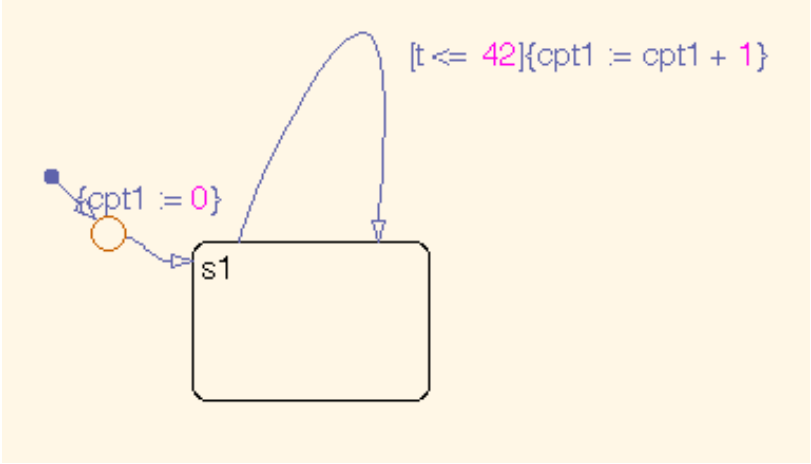


Select solver `ode45` (Dormand-Prince): we get Amp = 1; Freq = 1 (sinusoid).

Strange behaviours of Hybrid modelers

E.g., Simulink/Stateflow.





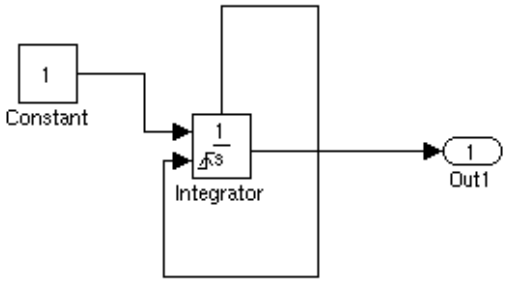
Select solver `ode45` (Dormand-Prince): we get Amp = 1; Freq = 5 (sinusoid).

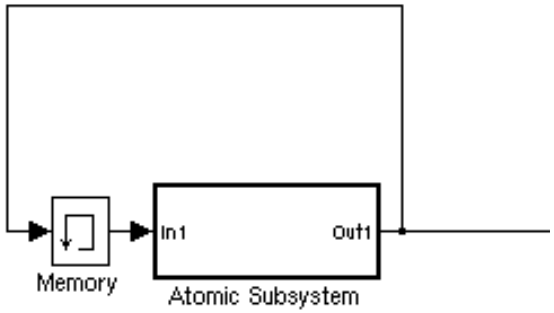
Changing the frequency/solver changes the semantics. Only one transition is taken during a step and it takes some amount of time.

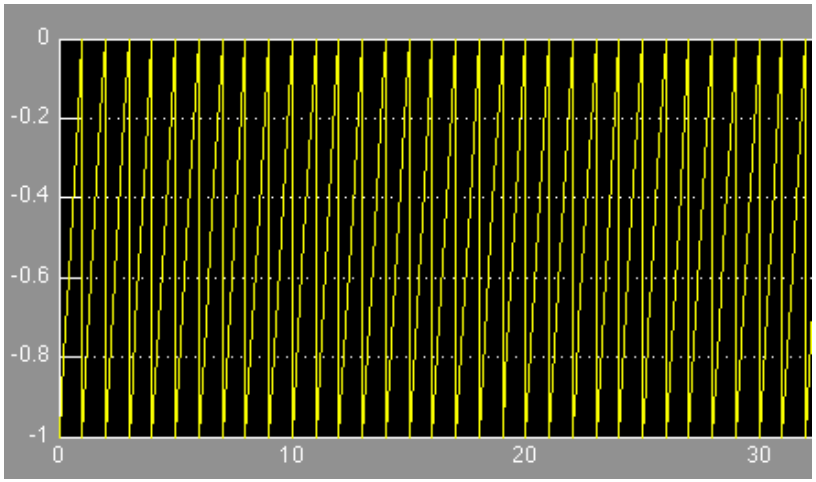
Strange behaviours of Hybrid modelers

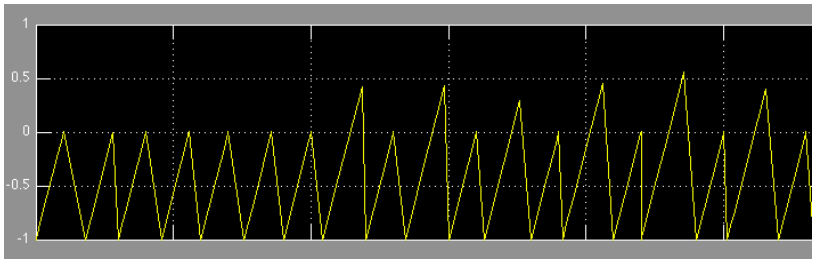
Adding delay/memory blocks or shared variables is sometimes mandatory to break algebraic loops.

E.g., use the state port when resetting an integrator.





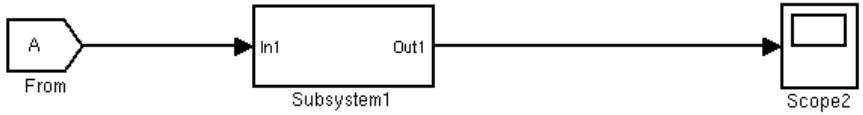


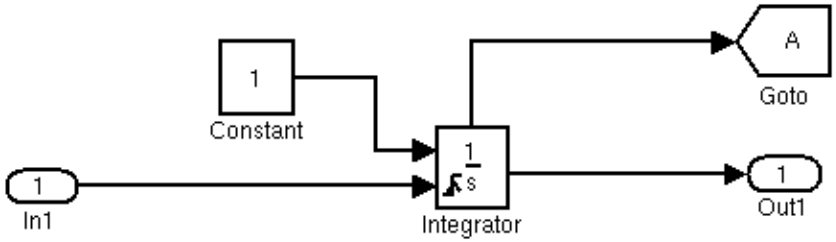


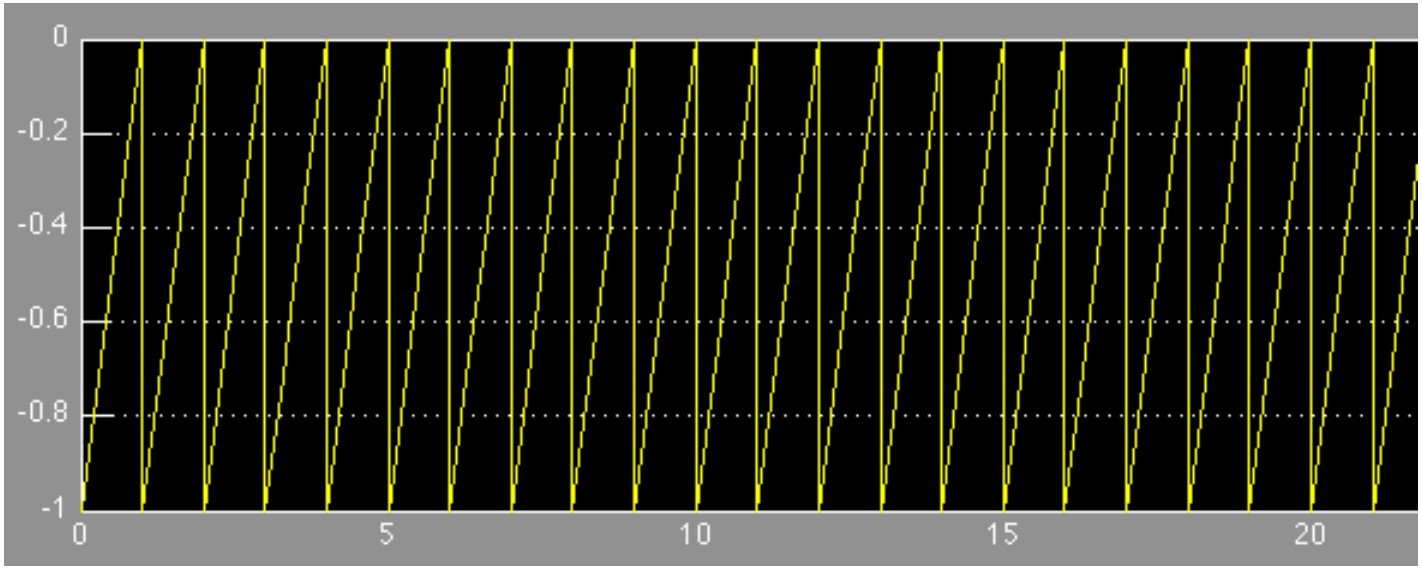
The stateport cannot be outputed. Yet, it is possible to store it into a write block. The memory block breaks the loop but changes the behaviour.

Strange behaviours of Hybrid modelers

Use a goto/from port or a write/read block. The main system reads from A. The subsystems outputs the value of the stateport into A.

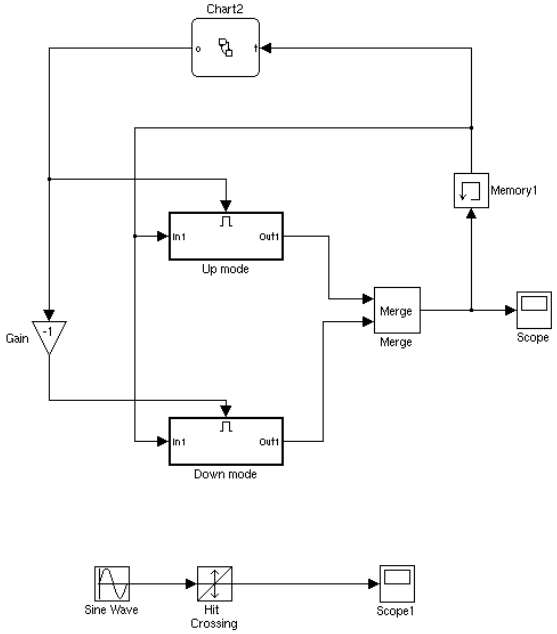


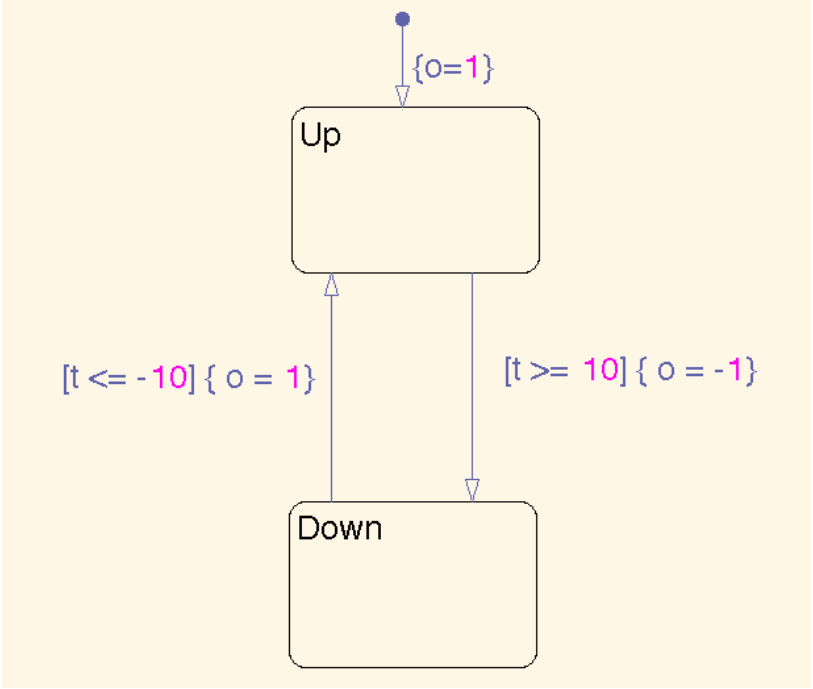


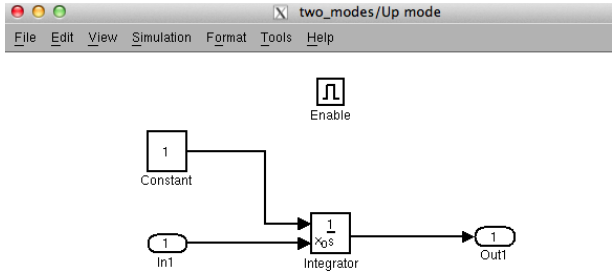


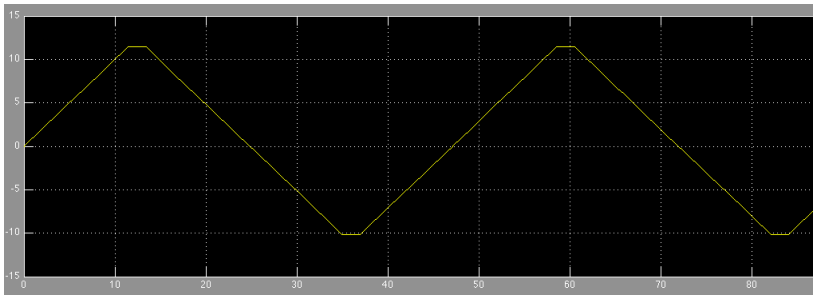
Strange behaviours of Hybrid modelers

Two teams, each of them in charge of a block (`Up` and `Down`). Merge them. A memory block is necessary to break the algebraic loop.



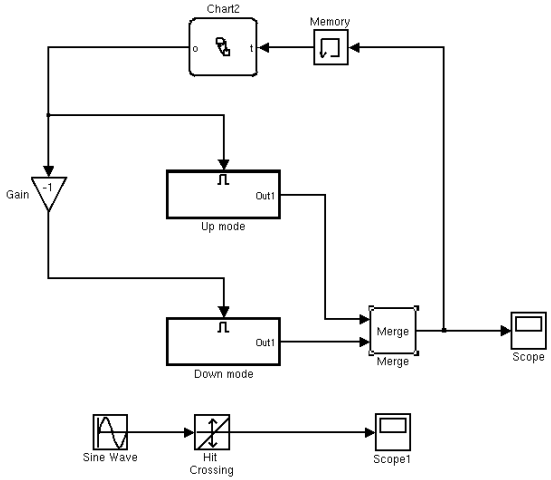


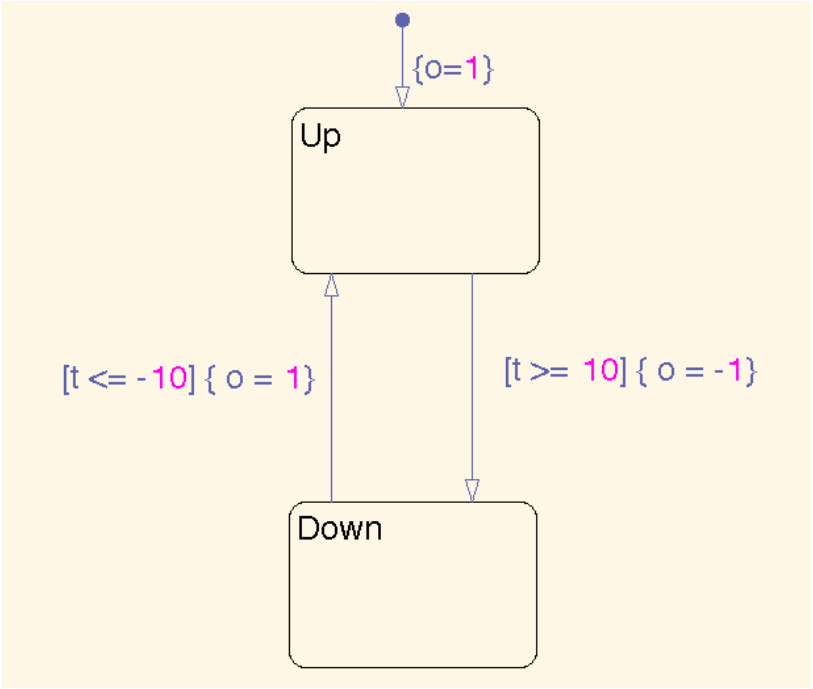


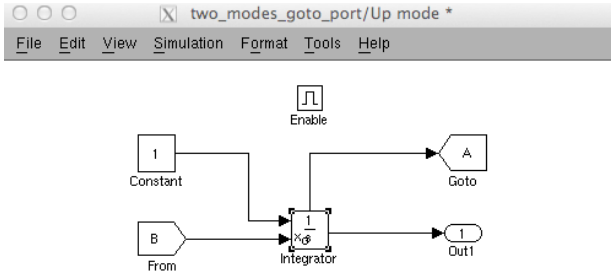


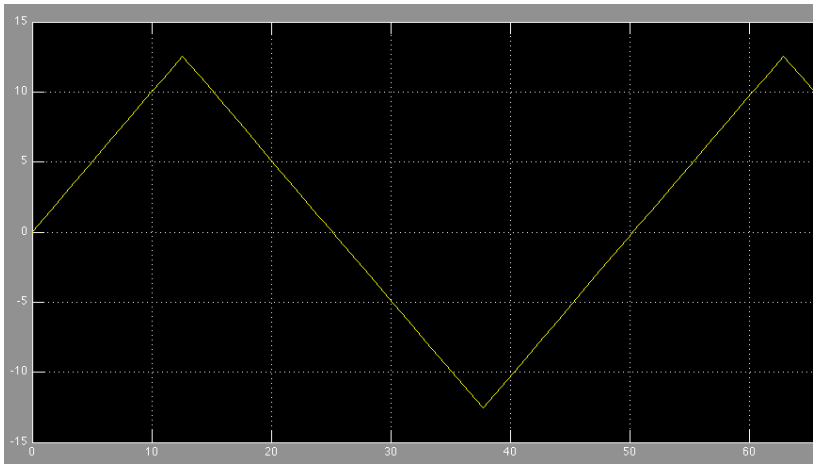
Strange behaviours of Hybrid modelers

Use the from/goto port to pass states between enabled subsystems. This work for two subsystems only. Yet, a memory block is necessary.



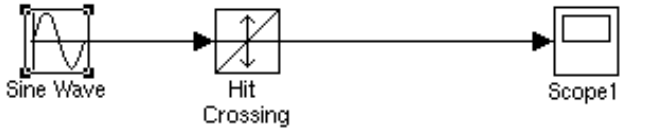


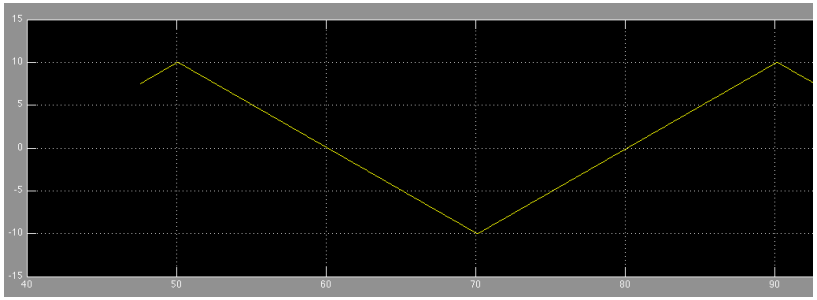


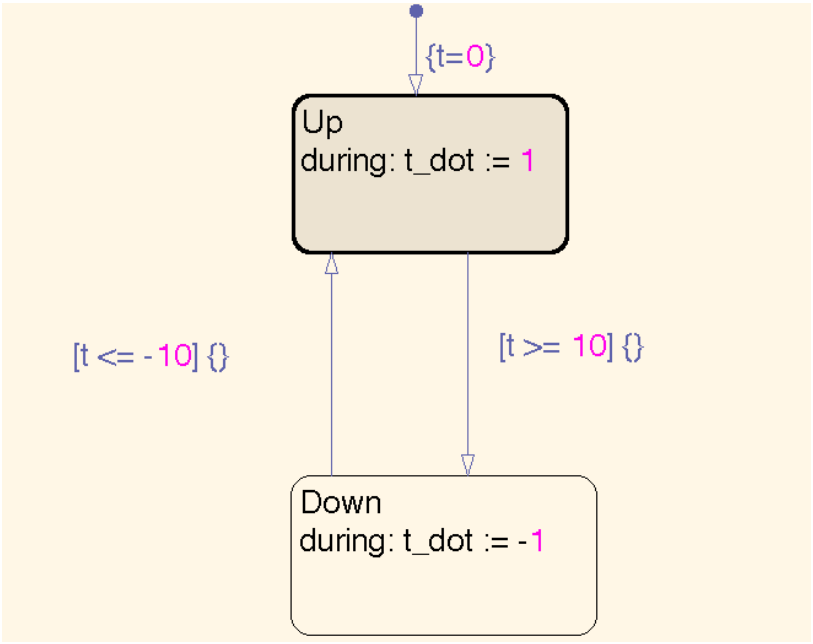


Strange behaviours of Hybrid modelers

Alternatively, directly write the hybrid automaton. But we have abandonned modular design (two different teams developing different parts of a model).







What went wrong?

The separation between discrete time and physical time is poorly made. Discrete time is the one of the global simulation.

Discrete time is here not logical time.

We propose the following discipline [LCTES'12]:

A signal is *discrete* if it is activated on a *discrete clock*, that is so defined:

> A clock is termed *discrete* if it has been declared so or if it is the result of a zero-crossing or a sub-sampling of a discrete clock. Otherwise, it is termed *continuous*.

This is not enough. Algebraic loops must be broken.

> Provide a construct `last` x that returns the previous value of x. In Non-standard semantics [JCSS'12], it coincides with the left-limit of x when x is left-continuous; the previous one otherwise.

These two disciplines can be ensured using static typing and causality analysis.

This way, logical time can be reconciled with continuous-time.

References

[1] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. A Hybrid Synchronous Language with Hierarchical Automata: Static Typing and Translation to Synchronous Code. In *ACM SIGPLAN/SIGBED Conference on Embedded Software (EMSOFT'11)*, Taipei, Taiwan, October 2011.

[2] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. Divide and recycle: types and compilation for a hybrid synchronous language. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES'11)*, Chicago, USA, April 2011.

[3] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. Non-Standard Semantics of Hybrid Systems Modelers. *Journal of Computer and System Sciences (JCSS)*, 78:877–910, May 2012. Special issue in honor of Amir Pnueli.

[4] Albert Benveniste, Benoit Caillaud, and Marc Pouzet. The Fundamentals of Hybrid Systems Modelers. In *49th IEEE International Conference on Decision and Control (CDC)*, Atlanta, Georgia, USA, December 15-17 2010.