CS:4980 Foundations of Embedded Systems

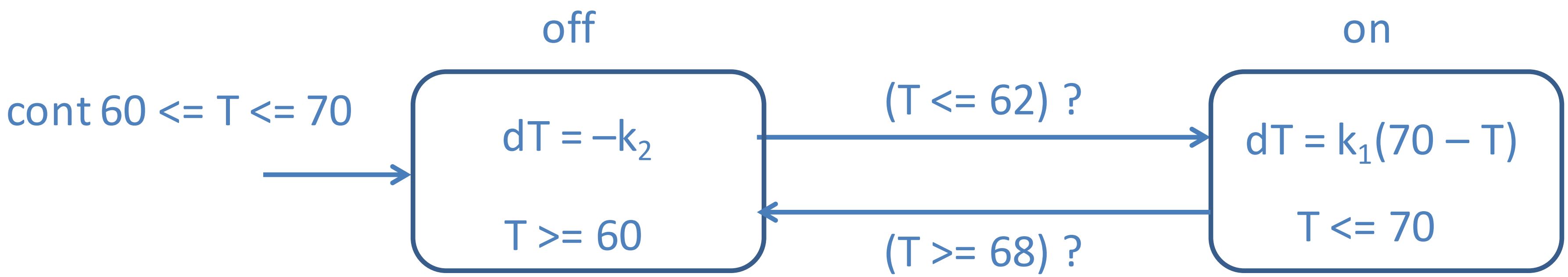
Hybrid Systems

Part I

Models of Reactive Computation

- Continuous-time model for dynamical system
  - Synchronous, where time evolves continuously
  - Execution of system: Solution to algebraic / differential equations
- Timed model
  - Like asynchronous for communication of information
  - Clocks evolve continuously, and constraints on delays allow synchronous/global coordination
- Hybrid systems
  - Generalization of timed processes
  - During timed transitions, evolution of state/output variables specified using differential equations as in dynamical systems

Self-Regulating Switching Thermostat



State machine with two modes: on and off

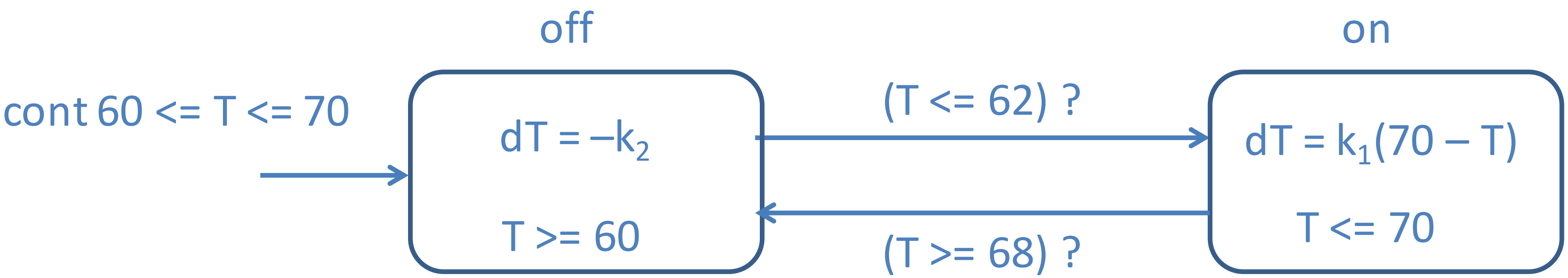
State variable T of type cont (continuous), to model temperature

T can be tested and updated during mode-switches

T changes continuously during timed transitions given by differential equations

Invariants (as in timed model) constrain how long can a timed transition be

Executions of Thermostat



Initial state = (off, T_0) with T_0 in the interval $[60,70]$

During a timed transition, T decreases continuously: $\mathbf{T}(t) = T_0 - k_2\, t$

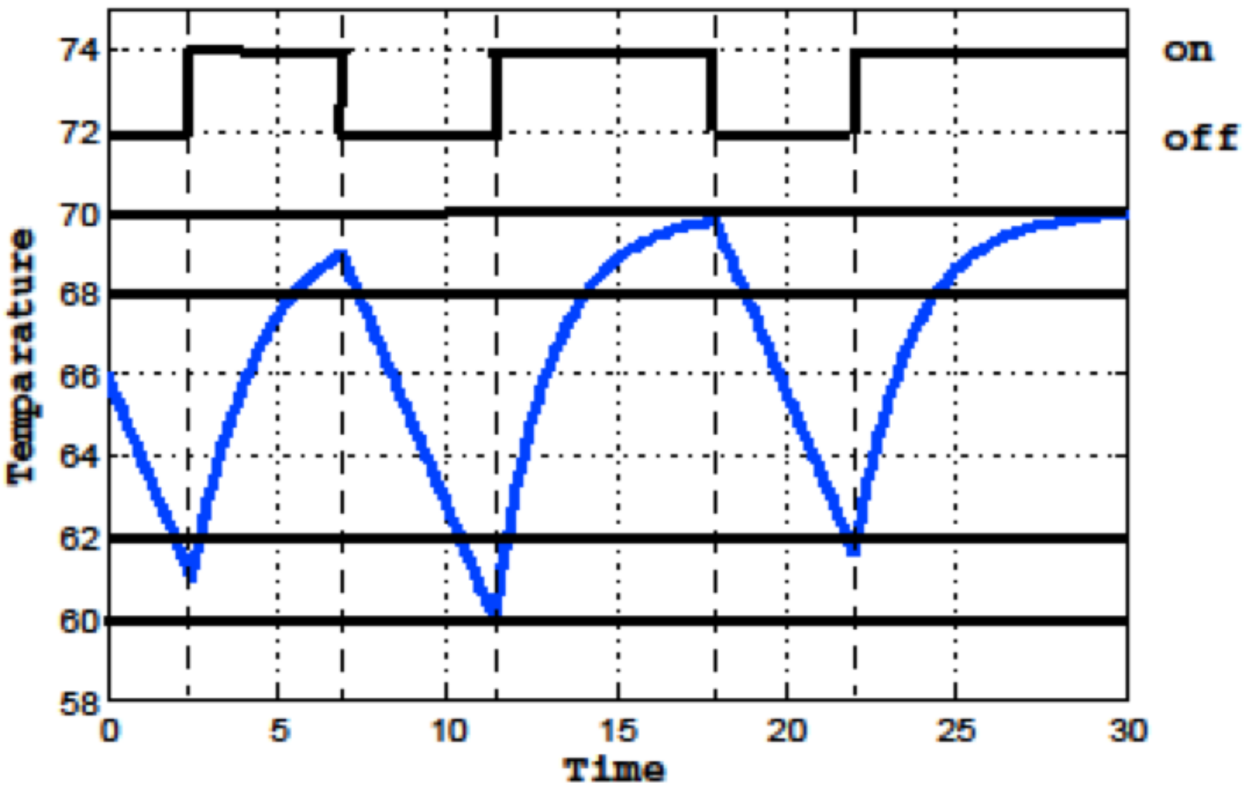
Mode-switch to on enabled when $T <= 62$, and must happen before T reaches 60

As time elapses in mode on, T increases according to $\mathbf{T}(t) = 70 - (70 - T^*)\, e^{-k1(t-t^*)}$

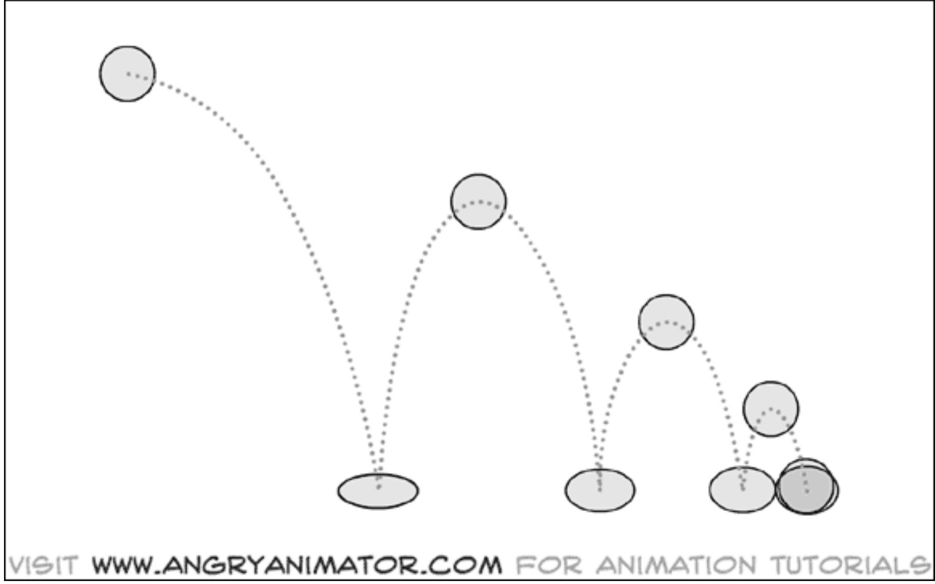
t^*, T^* : time and temperature upon entry to mode on

Mode-switch to off enabled when $T >= 68$, and must happen before T reaches 70

Simulation Plot of an Execution

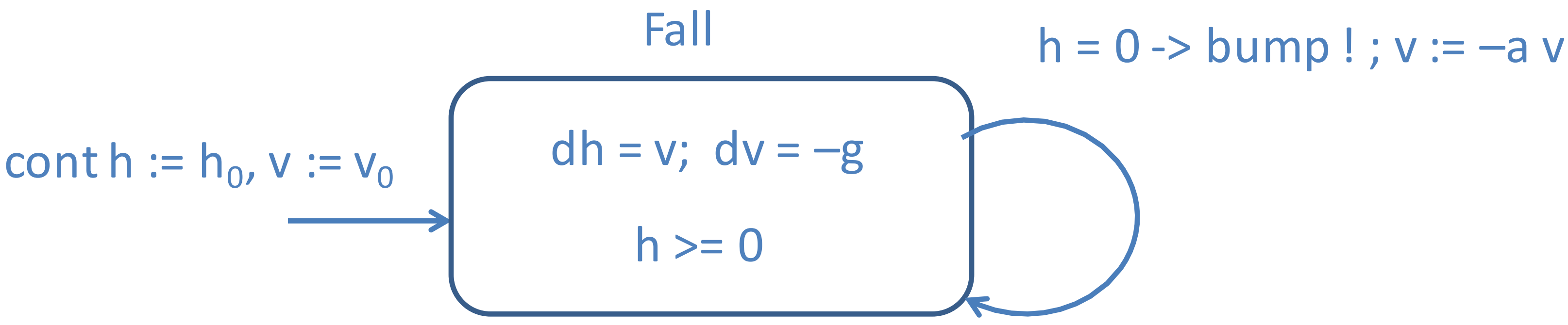


Modeling a Bouncing Ball



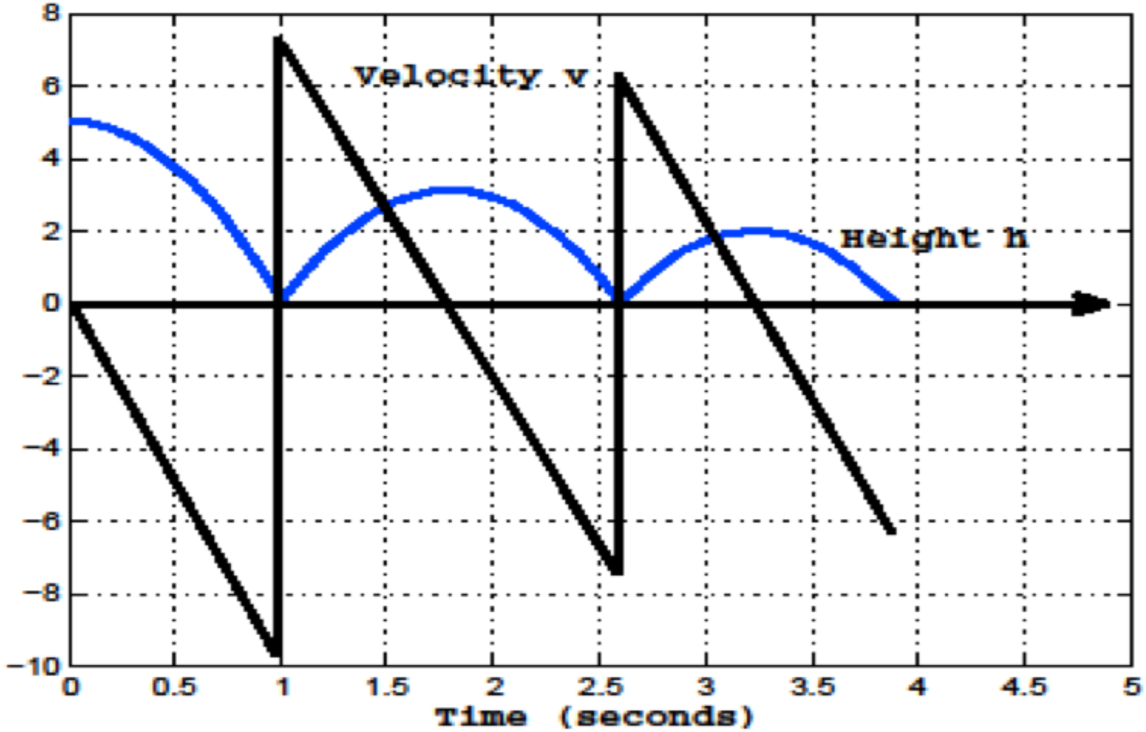
- Ball dropped from an initial height h_0 with an initial velocity v_0
- Velocity changes according to the differential equation $dv/dt = -g$
- When the ball hits the ground, that is, when height $h = 0$, velocity changes discretely: $v := - a\ v$, where $0 < a < 1$ is *dampening* constant
- Modeled as a hybrid system: mix of discrete and continuous behaviors!

Hybrid Process for Bouncing Ball



Execution of the Bouncing Ball Process

$h_0 = 5$
$v_0 = 0$



Definition of Hybrid Process: Syntax

A *hybrid process* HP consists of

1. An asynchronous process P, with continuous (I_c) and discrete (I_d) input variables I, continuous (S_c) and discrete (S_d) state variables S, and continuous (O_c) and discrete (O_d) output variables O
2. A continuous-time invariant CI, a Boolean expression over S_c
3. For every continuous output variable y, a Lipschitz-continuous real-valued expression $h_y(S_c, I_c)$ defining y
4. For every continuous state variable x, a Lipschitz-continuous real-valued expression $f_x(S_c, I_c)$ defining the rate of change of x
5. Input, output, internal and timed actions

Definition of Hybrid Process: Semantics

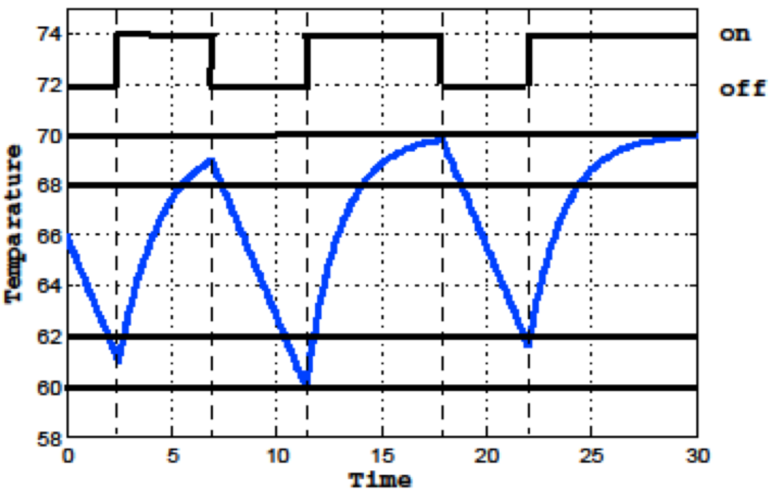
- Inputs, outputs, states, initial states, internal actions, input actions, output actions: defined exactly the same as the asynchronous model
- Timed actions: Given a state s and real-valued time $\delta > 0$ and a continuous input signal $\mathbf{I}(t)$ giving values for I_c over time interval $[0, \delta]$, the corresponding state signal $\mathbf{S}(t)$ and output signal $\mathbf{O}(t)$ over $[0, \delta]$ is uniquely defined so that
  1. The initial state $\mathbf{S}(0)$ equals s
  2. For each output variable y in O_c, $\mathbf{O}_y(t) = h_y(\mathbf{S}(t), \mathbf{I}(t))$
  3. For each state variable x in O_c, $d\mathbf{S}_x(t)/dt = f_x(\mathbf{S}(t), \mathbf{I}(t))$
  4. At all times t in $[0, \delta]$, $\mathbf{S}(t)$ satisfies the invariant CI

  Note: At all times t in $[0, \delta]$, discrete state variables stay unchanged

Executions of Hybrid Processes

Starting from an initial state, execute either

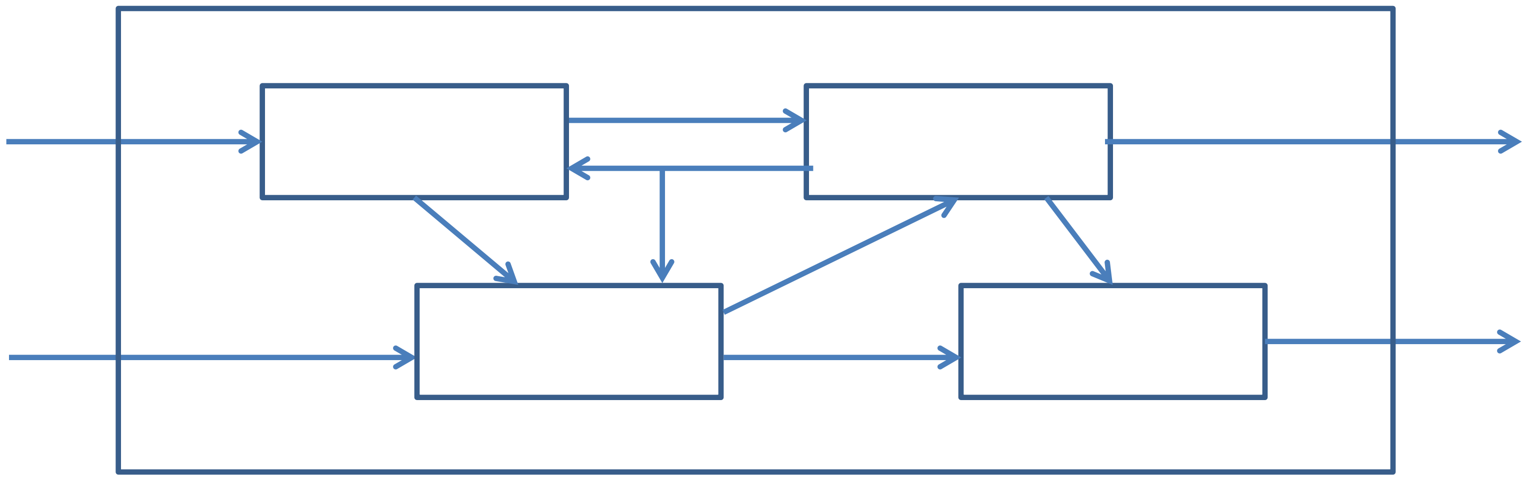
- a discrete, instantaneous step (input, or output or internal action) or
- a timed step of some duration $\delta > 0$ (need to solve system of ODEs)



(off, 66) -2.5-> (off, 61) -> (on, 61) -3.7-> (on,69.02) -> (off, 69.02) -4.4-> (off, 60.22) -> (on, 60.22) -7.6-> (on, 69.9) -> (off, 69.9) -4.1-> (off, 61.7) -> (on, 61.7) …

Concepts based on transition systems such as reachable states, safety and liveness requirements, all apply to hybrid systems

Block Diagrams



- ❑ Component processes can now be hybrid processes
  - Need to define instantiation, composition, output hiding

- ❑ Channels connecting processes of two types
  1. Sender/receiver communication during discrete steps: as in the asynchronous model
  2. Continuously evolving signals during timed steps: as in the model of continuous-time dynamical systems

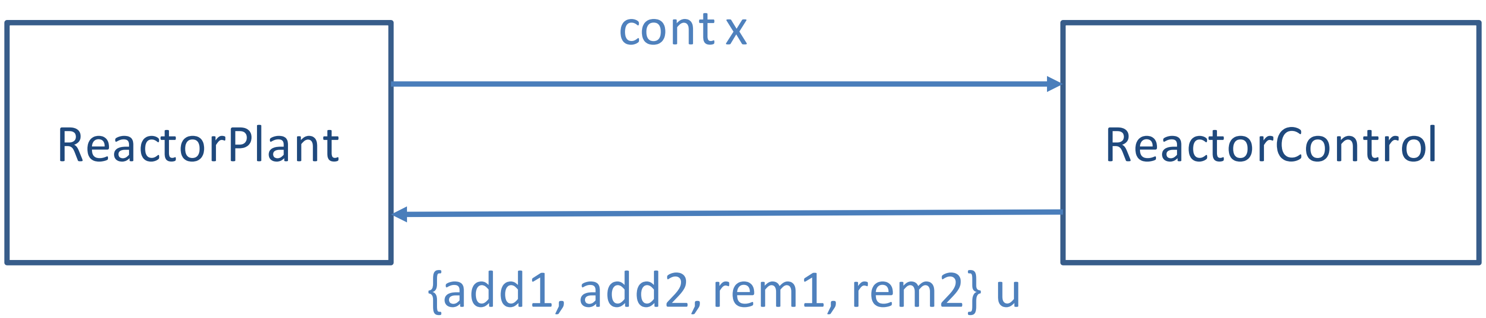
Composition of Hybrid Processes

- Instantiation, variable renaming and output hiding:
  - Defined as usual

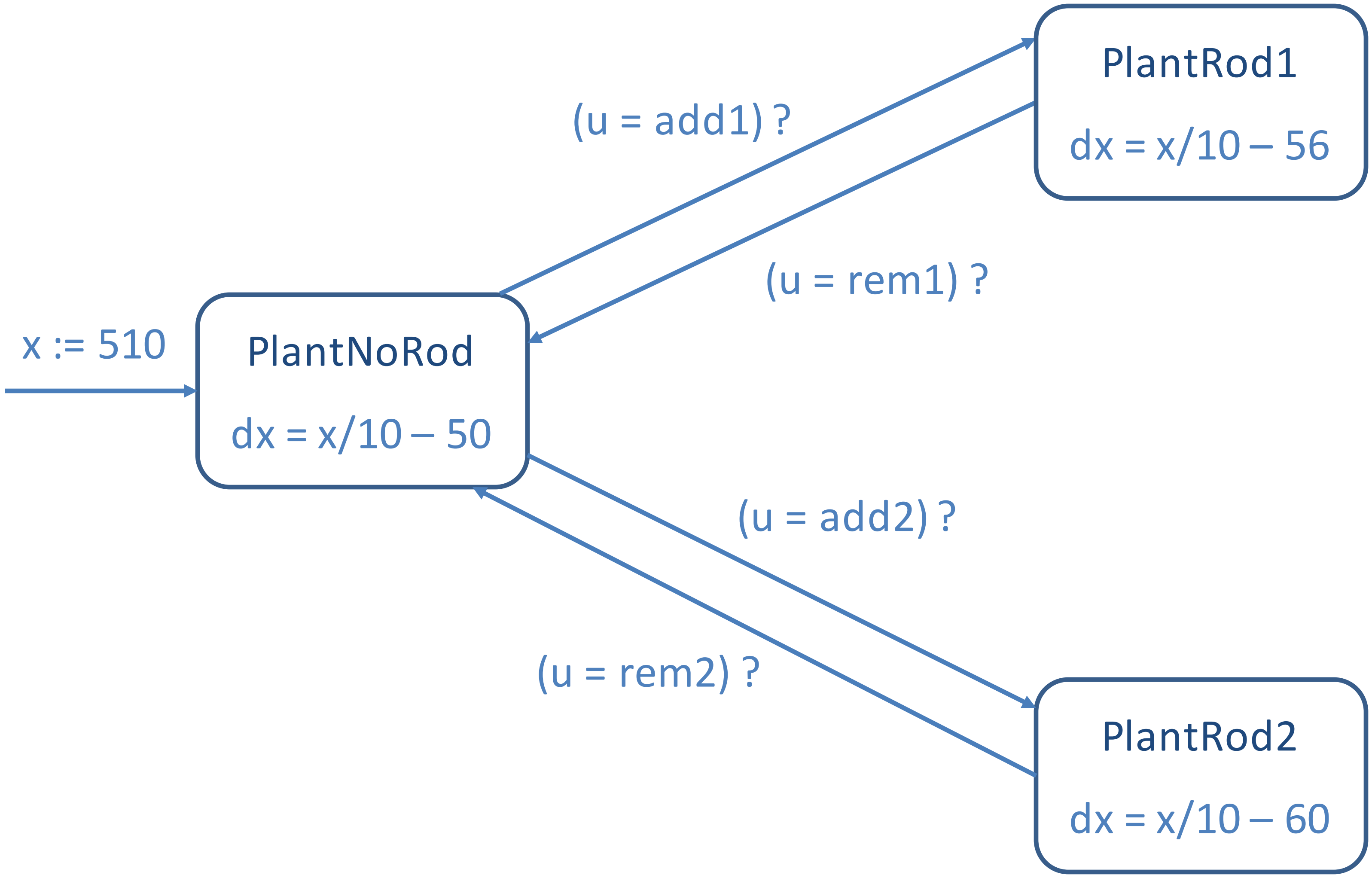
- Composition:
  - Compose discrete parts together as in the asynchronous model
  - Compose continuous parts of internal actions together as in dynamical systems
  - Generate continuous-time invariants of merged action a conjunction of original ones

- Compatibility of two hybrid processes:
  - State variables are disjoint and output variables are disjoint
  - No cyclic await dependencies among shared input/output variables

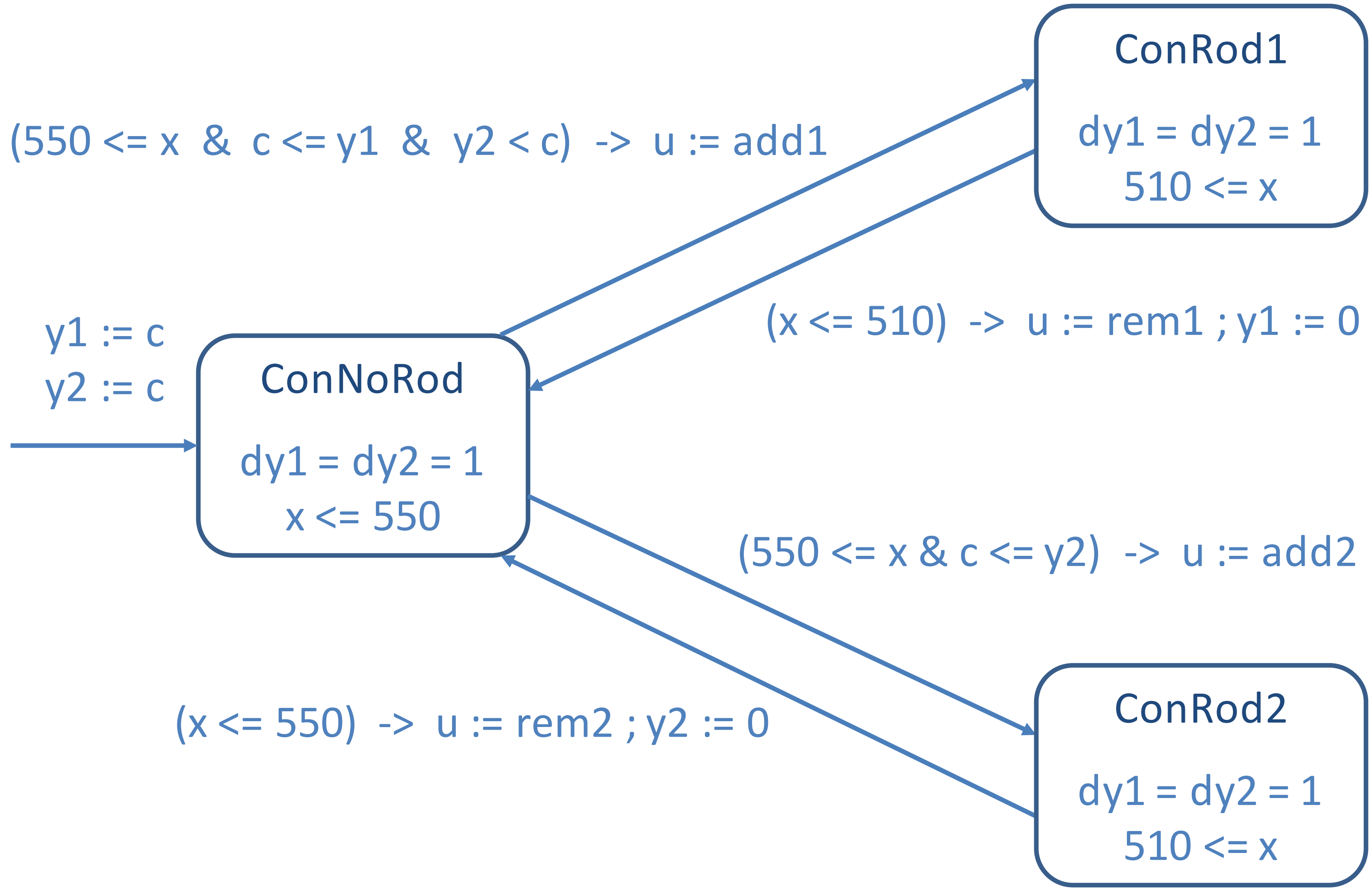
Nuclear Reactor Example



Reactor Plant



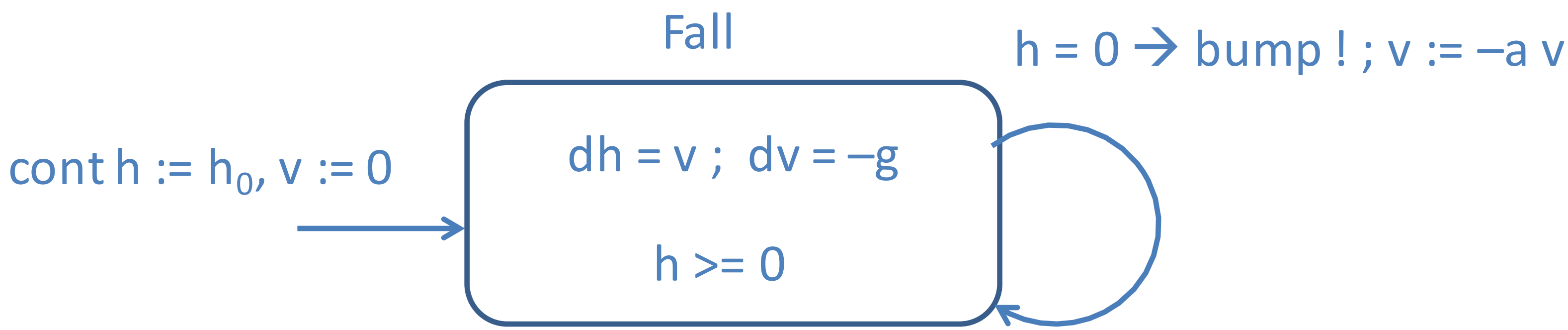
Reactor Controller



Summary of the Model

- Generalizes timed model
  - Variables evolving continuously during a timed action can have complex dynamics, clocks being a very special case
- Generalizes continuous-time dynamical systems
  - Discontinuous changes to system state now can be modeled
- Generalizes asynchronous model
  - Distributed/multi-agent systems can be modeled
- Suitable for modeling of cyber-physical systems (in full generality)
- Existing commercial tool support: Modelica, Stateflow/Simulink
- Challenge for analysis
  - Even if dynamics in individual modes is linear, due to discrete changes it is not possible to obtain closed-form solutions, or general theorems about stability

Analysis of Bouncing Ball Model



Change in height during first bounce: $\mathbf{h}(t) = h_0 - gt^2/2$

Time at which first bump occurs: t_1 = Sqrtr $(2\, h_0 / g)$

Velocity just before first bump occurs: –Sqrt $(2\, g\, h_0) = -v_1$

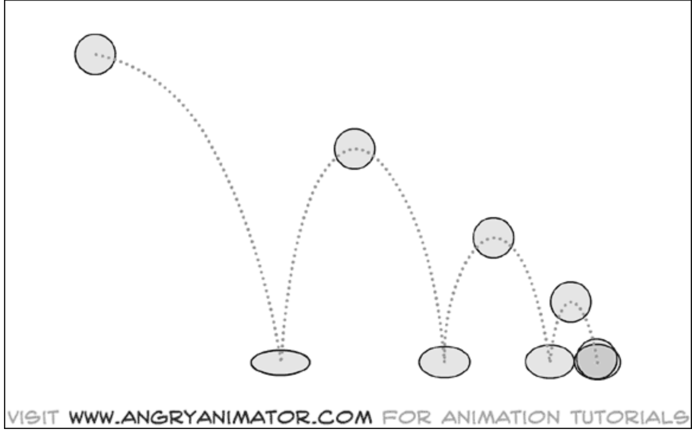
Velocity just after first bump : $v_2 = a\, v_1$

Evolution of height during second bounce: $\mathbf{h}(t) = v_2\, t - gt^2/2$

Time between first and second bump: $t_2 = 2\, v_2 / g$

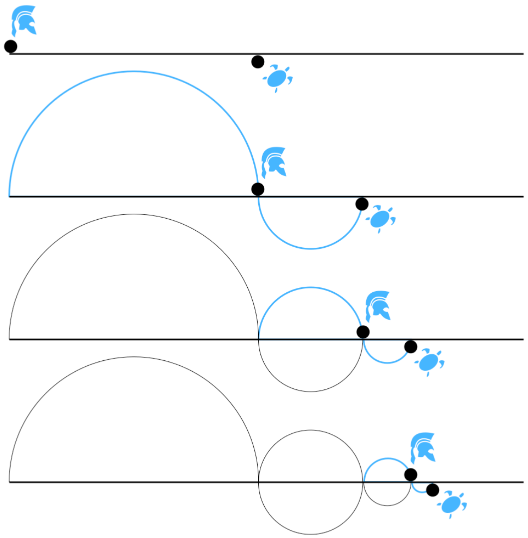
Velocity just before second bump occurs: $-v_2$ and after 2nd bump $v_3 = a\, v_2$

Modeling a Bouncing Ball



- Velocity after k bumps: $a^k v_1$
- Duration between k^{th} and following bump: $a^k v_1 / g$
- Sum of durations between successive bumps converges to $v_1 (1 + a) / (1 - a)$
- Infinitely many discrete actions in finite time: Zeno behavior!

Zeno's Paradox



- Described by Greek philosopher Zeno in context of a race between Achilles and a tortoise
- Tortoise has a head start over Achilles, but is much slower
- In each discrete round, suppose Achilles is d meters behind at the beginning of the round
- During the round, Achilles runs d meters, but by then, tortoise has moved a little bit further
- At the beginning of the next round, Achilles is still behind, by a distance of $a\, d$ meters, with $0 < a < 1$
- By induction, if we repeat this for infinitely many rounds, Achilles will never catch up!
- If the sum of durations between successive discrete actions converges to a constant K, then an execution with infinitely many discrete actions describes behavior only up to time K (and does not tell us the state of the system at time K and beyond)

Formalization

- An infinite execution of a hybrid process HP is of the form s_0 -t_1-> s_1 -t_2-> s_2 -t_3-> s_3 …, where t_i is the duration of i^{th} step
  - Input/output/internal actions are instantaneous (duration 0)

- An infinite execution is called
  - *Zeno* if the infinite sum of all the durations is bounded above by a constant, and
  - *non-Zeno* if the sum diverges

- A state s of the process HP is called
  - *Zeno* if every execution starting in state s is Zeno
  - *Non-Zeno* if there is some non-Zeno execution starting in s

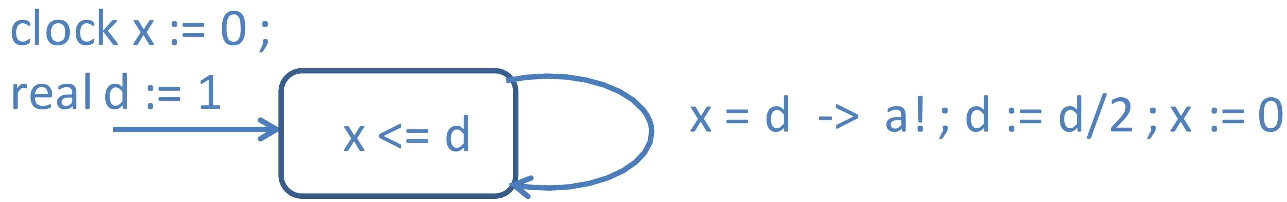
Formalization

- A hybrid process HP is called *non-Zeno* if every reachable state of HP is non-Zeno
  - At every point during an execution it is possible for time to diverge

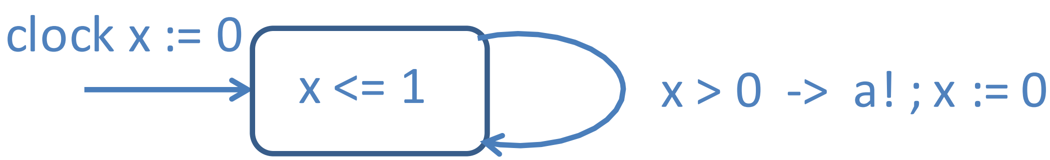
- Zeno system: Could end up in a state from which duration between successive steps must get smaller and smaller

- Examples
  - Thermostat: non-Zeno
  - Bouncing ball: Zeno

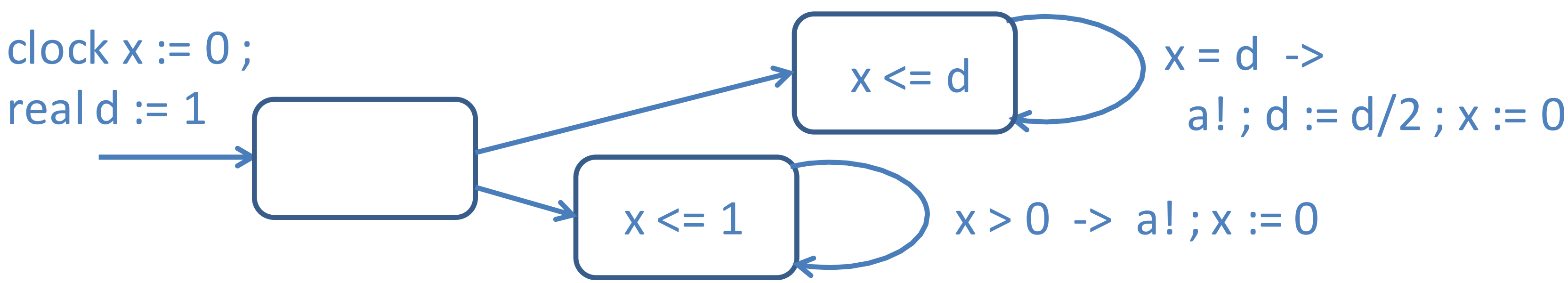
Zeno vs Non-Zeno



Zeno! Every possible execution is Zeno



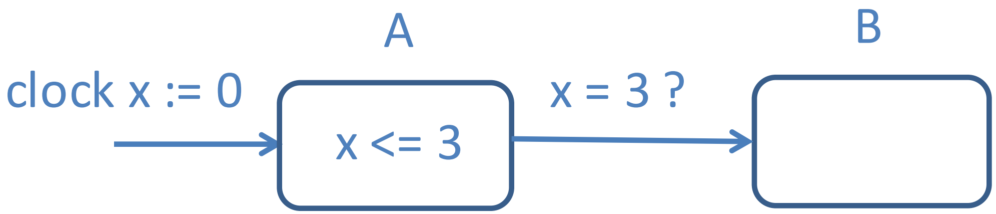
Non-Zeno! Some executions are Zeno and some are non-Zeno



Zeno! System may end up in a state from which only Zeno executions are possible

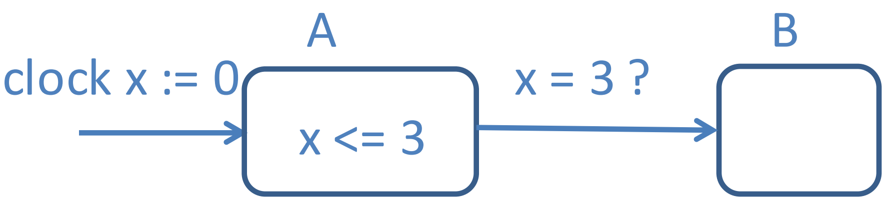
Zeno Processes and Reachability

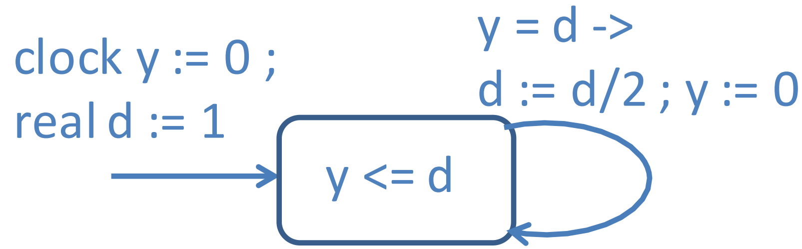
- How does existence of Zeno processes influence analysis?
- Recall:
  - A state s of a system H is *reachable* if there exists a finite execution starting in an initial state and ending in state s
  - A property P is *invariant* for H all reachable states satisfy P



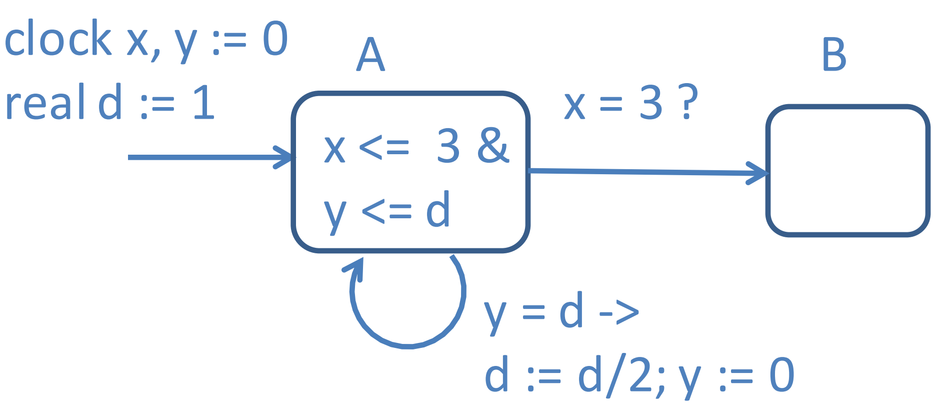
Is mode B reachable ?

Zeno Processes and Reachability



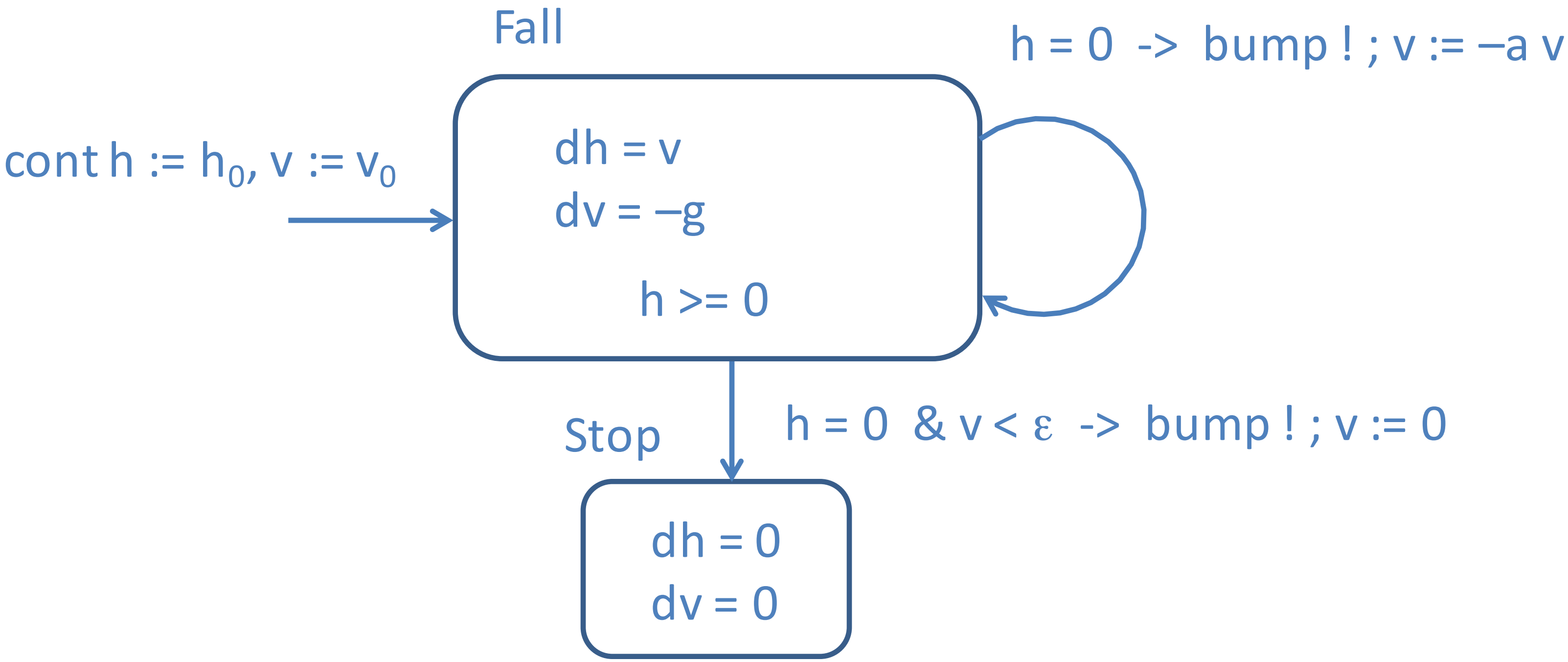


Is mode B reachable ?



Presence of a Zeno process in the system can stop time from advancing, and make states of other processes unreachable !

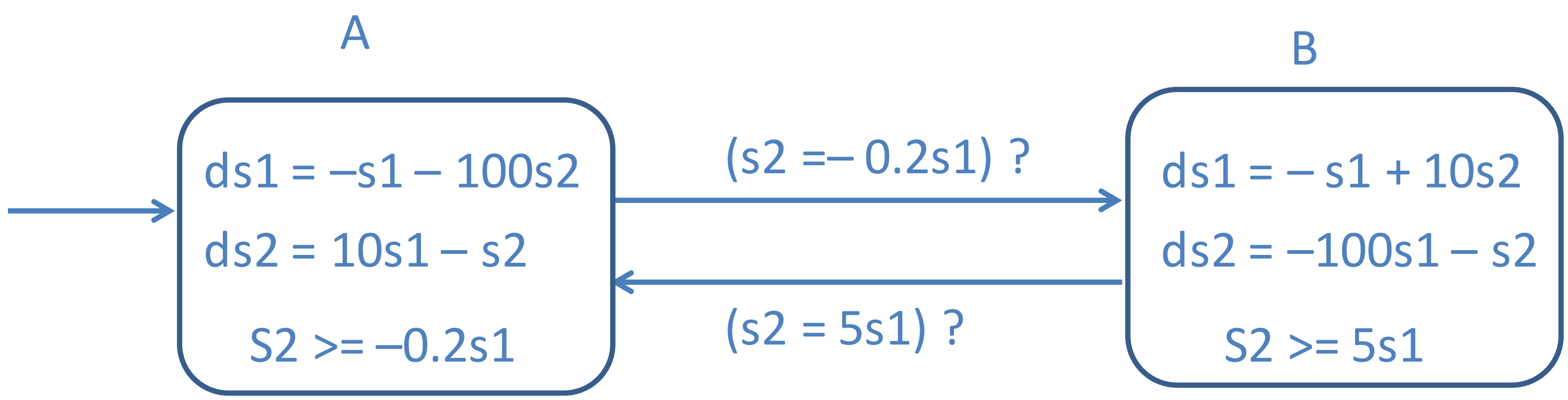
Making Bouncing Ball Non-Zeno



If velocity is too small, stop modeling dynamics accurately

In this model, there is a lower bound on duration between successive bumps

Stability of Hybrid Systems

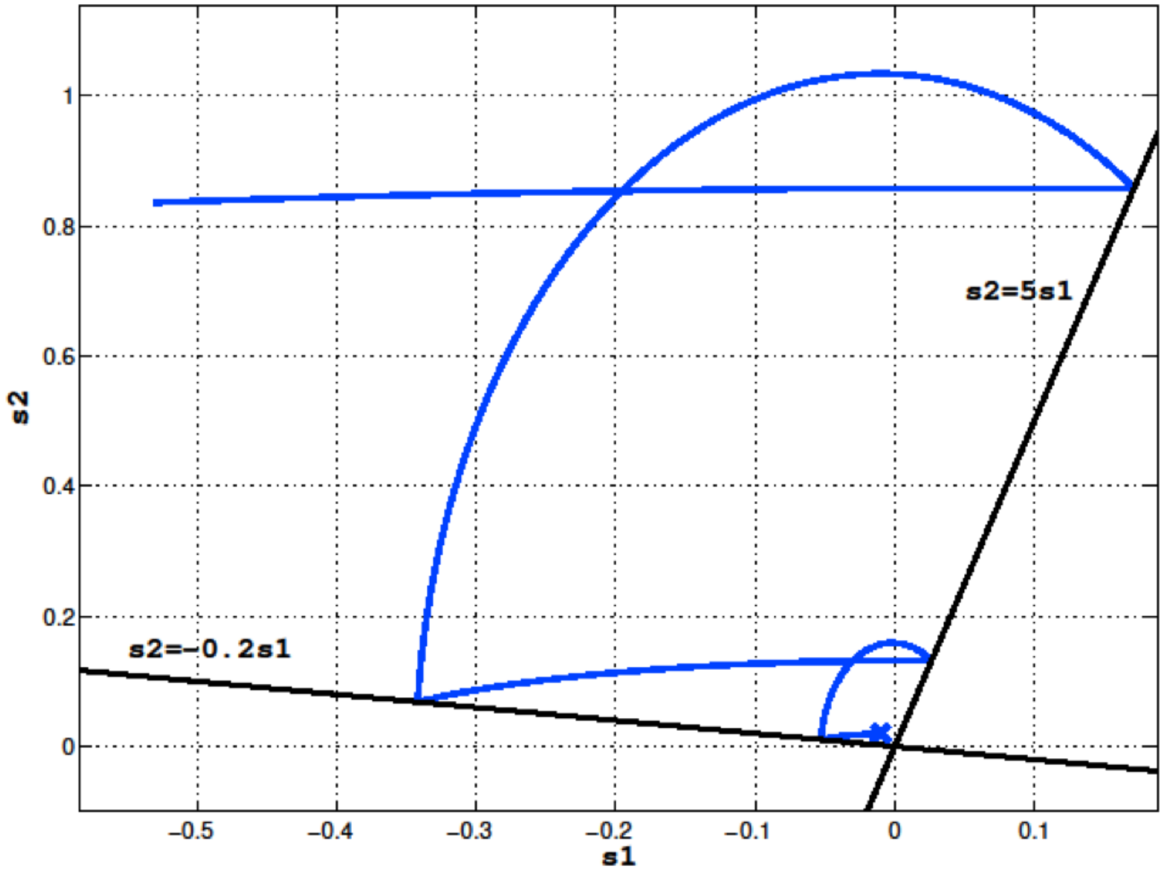


Is the dynamics in mode A stable?

Is the dynamics in mode B stable?

Each mode has stable dynamics, but switching causes instability!

Stability of Hybrid Systems



Credits

Notes based on Chapter 9 of

Principles of Cyber-Physical Systems
by Rajeev Alur
MIT Press, 2015