

Operational Semantics of Hybrid Systems

Edward A. Lee and Haiyang Zheng

With contributions from:

Adam Cataldo, Jie Liu, Xiaojun Liu,
Eleftherios Matsikoudis

Chess Seminar, November 16, 2004

Abstract

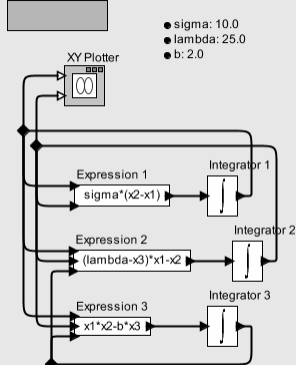


In this talk, I will discuss an interpretation of hybrid systems as executable models. A specification of a hybrid system for this purpose can be viewed as a program in a domain-specific programming language. The semantic properties of that programming language have considerable bearing on the understandability, executability, and analyzability of a model. I will discuss several semantic issues that come up when attempting to define such a programming language, including the consequences of numerical ODE solver techniques, the interpretation of discontinuities in continuous-time signals, and the interpretation of discrete-event signals.

Basic Continuous-Time Modeling



Continuous-Time (CT) Solver

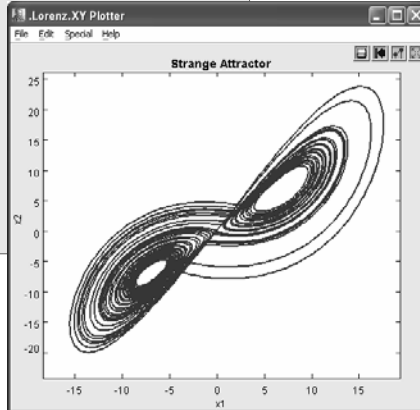


Author: Jie Liu

- sigma: 10.0
- lambda: 25.0
- b: 2.0

This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

A basic continuous-time model describes an ordinary differential equation (ODE).

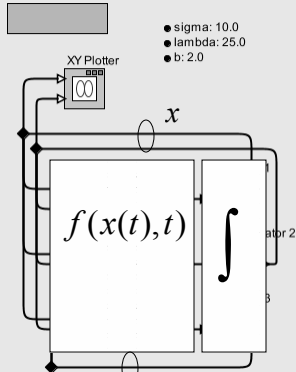


Lee, Hybrid: 3

Basic Continuous-Time Modeling



Continuous-Time (CT) Solver



Author: Jie Liu

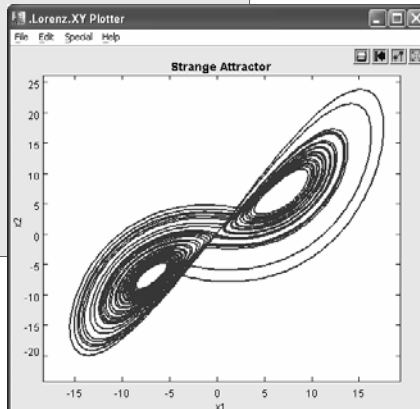
- sigma: 10.0
- lambda: 25.0
- b: 2.0

This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

A basic continuous-time model describes an ordinary differential equation (ODE).

$$\dot{x}(t) = f(x(t), t)$$

$$x(t) = x(t_0) + \int_{t_0}^t \dot{x}(\tau) d\tau$$



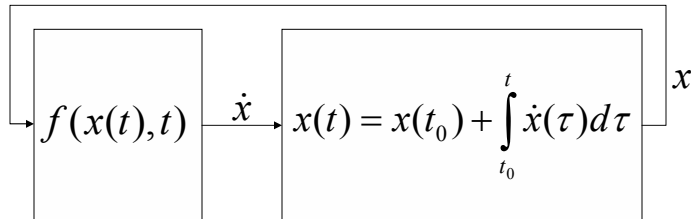
Lee, Hybrid: 4

Basic Continuous-Time Modeling



The state trajectory is modeled as a vector function of time,

$$x: T \rightarrow R^n \quad T = [t_0, \infty) \subset R$$



$$\dot{x}(t) = f(x(t), t)$$

$$f: R^m \times T \rightarrow R^m$$

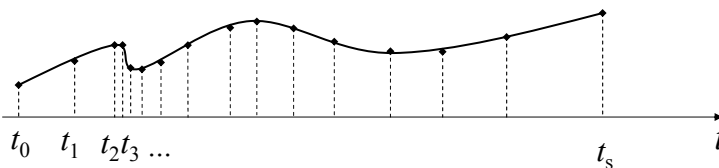
Lee, Hybrid: 5

ODE Solvers



Numerical solution approximates the state trajectory of the ODE by estimating its value at discrete time points:

$$\{t_0, t_1, \dots\} \subset T$$



Reasonable choices for these points depend on the function f .

Lee, Hybrid: 6

E.g. Runge-Kutta 2-3 Solver (RK2-3)



Given $x(t_n)$ and a time increment h , calculate

$$\begin{aligned}
 K_0 &= f(x(t_n), t_n) \quad \leftarrow \dot{x}(t_n) \\
 K_1 &= f(x(t_n) + 0.5hK_0, t_n + 0.5h) \quad \leftarrow \text{estimate of } \dot{x}(t_n + 0.5h) \\
 K_2 &= f(x(t_n) + 0.75hK_1, t_n + 0.75h) \quad \leftarrow \text{estimate of } \dot{x}(t_n + 0.75h)
 \end{aligned}$$

then let

$$t_{n+1} = t_n + h$$

$$x(t_{n+1}) = x(t_n) + (2/9)hK_0 + (3/9)hK_1 + (4/9)hK_2$$

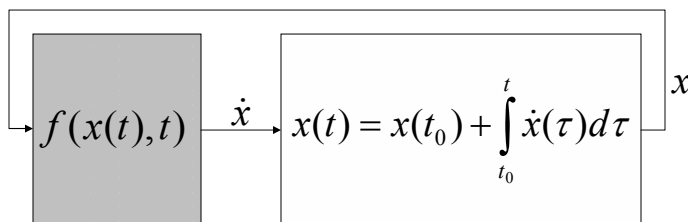
Note that this requires three evaluations of f at three different times with three different inputs.

Lee, Hybrid: 7

Operational Requirements



In a software system, the blue box below can be specified by a program that, given $x(t)$ and t calculates $f(x(t), t)$. But this requires that the program be functional (have no side effects).



$$\dot{x}(t) = f(x(t), t)$$

$$f : R^m \times T \rightarrow R^m$$

Lee, Hybrid: 8

Adjusting the Time Steps



For time step given by $t_{n+1} = t_n + h$, let

$$K_3 = f(x(t_{n+1}), t_{n+1})$$

$$\varepsilon = h((-5/72)K_0 + (1/12)K_1 + (1/9)K_2 + (-1/8)K_3)$$

If ε is less than the “error tolerance” e , then the step is deemed “successful” and the next time step is estimated at:

$$h' = 0.8 \sqrt[3]{e/\varepsilon}$$

If ε is greater than the “error tolerance,” then the time step h is reduced and the whole thing is tried again.

Lee, Hybrid: 9

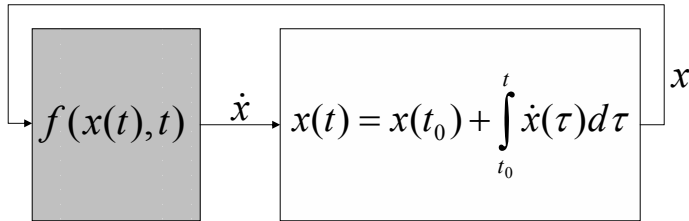
Does this Make You Feel Sick?



No wonder Alessandro Abate wants to model this as a stochastic system!

Lee, Hybrid: 10

Examining This Computationally



At each discrete time t_n , given a time increment $t_{n+1} = t_n + h$, we can estimate $x(t_{n+1})$ by repeatedly evaluating f with different values for the arguments. We may then decide that h is too large and reduce it and redo the process.

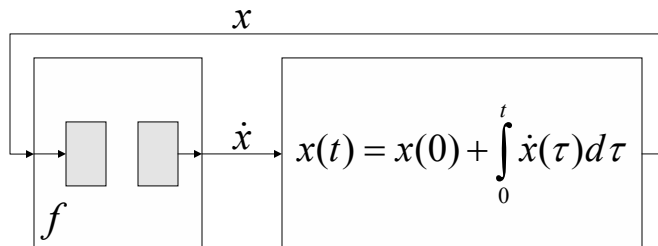
Lee, Hybrid: 11

How General Is This Model?



Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines?



$$\dot{x}(t) = f(x(t), t)$$

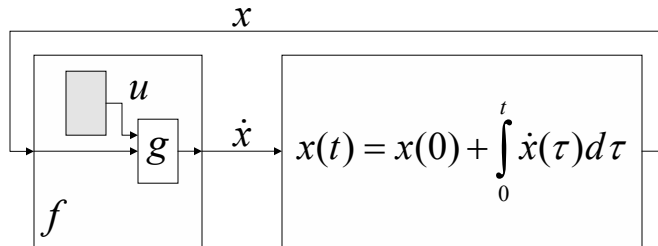
Lee, Hybrid: 12

How General Is This Model?



Does it handle:

- Systems without feedback?
- External inputs? yes
- State machines?



$$\dot{x}(t) = f(x(t), t) = g(u(t), x(t), t)$$

Lee, Hybrid: 13

The Model Itself as a Function



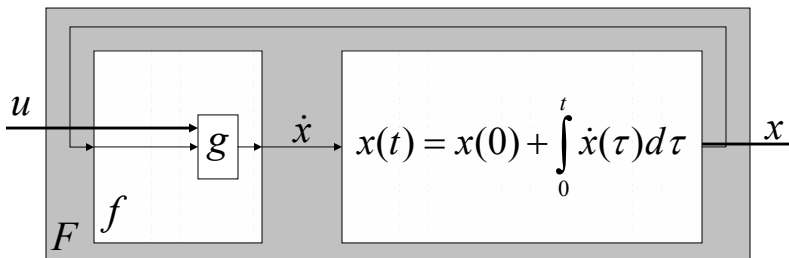
Note that the model function has the form:

$$F : [T \rightarrow R^m] \rightarrow [T \rightarrow R^m]$$

Which does not match the form:

$$f : R^m \times T \rightarrow R^m$$

Set of functions



(This assumes certain technical requirements on f and u that ensure existence and uniqueness of the solution.)

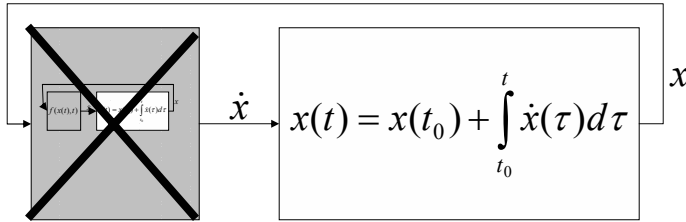
Lee, Hybrid: 14

Consequently, the Model is Not Compositional!



In general, the behavior of the inside dynamical system cannot be given by a function of form:

$$f : R^m \times T \rightarrow R^m$$



To see this, just note that the output must depend only on the current value of the input and the time to conform with this form.

Lee, Hybrid: 15

Aside



It is very common in control systems literature to write:

$$\dot{x} = f(x, t)$$

when what is meant is:

$$\dot{x}(t) = f(x(t), t)$$

RK2-3 assumes this form

These are not the same. Note:

$$\dot{x}, x \in [T \rightarrow R^m]$$

$$\dot{x}(t), x(t) \in R^m$$

So what are the domain and codomain of f ?

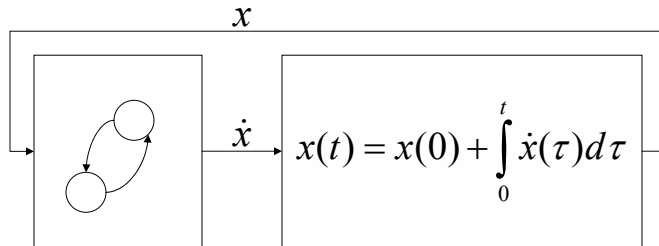
Lee, Hybrid: 16

So How General Is This Model?



Does it handle:

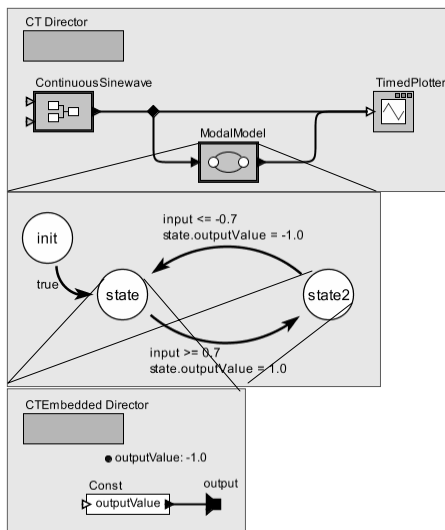
- External inputs?
- Systems without feedback?
- State machines? No... The model needs work...



Since this model is itself a state machine, the inability to put a state machine in the left box explains the lack of composability.

Lee, Hybrid: 17

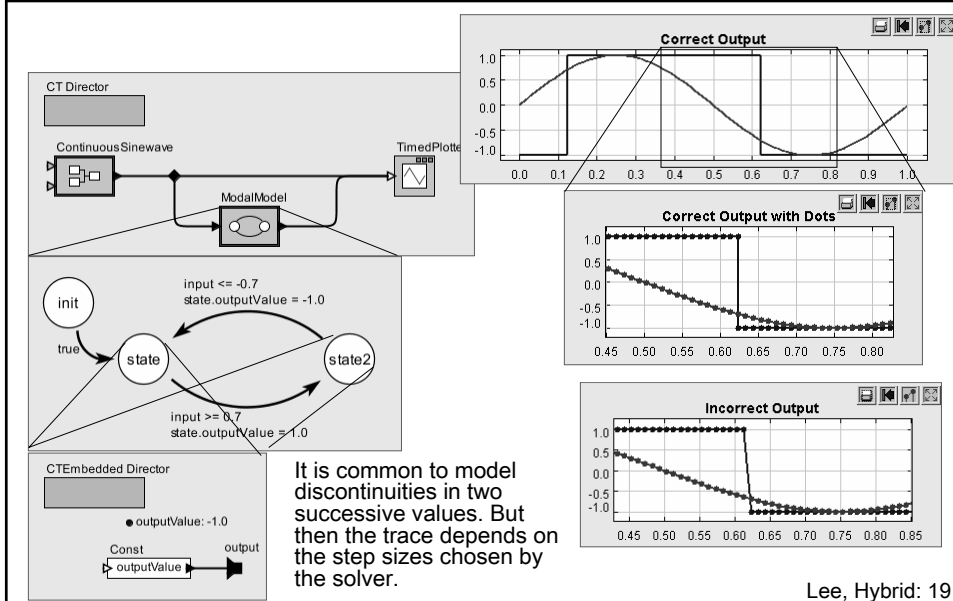
Start with Simple State Machines Hysteresis Example



This model shows the use of a two-state FSM to model hysteresis. Semantically, the output of the ModalModel block is discontinuous. If transitions take zero time, this is modeled as a signal that has two values *at the same time*, and *in a particular order*.

Lee, Hybrid: 18

Hysteresis Example



Requirements



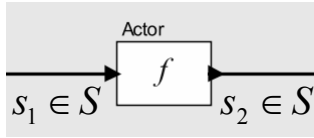
The hysteresis example illustrates two requirements:

- A signal may have more than one value at a particular time, and the values it has have an order.
- The times at which the solver evaluates signals must precisely include the times at which interesting events happen, like a guard becoming true.

Both Requirements Are Dealt With By an Abstract Semantics



Previously

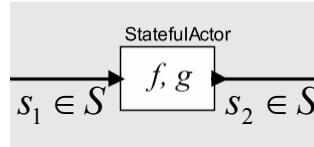


$$S = [T \rightarrow R]$$

$$f : R^m \times T \rightarrow R^m$$

$$\forall t \in T, s_2(t) = f(s_1(t), t)$$

Now we need:



$$S = [T \times N \rightarrow R]$$

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$

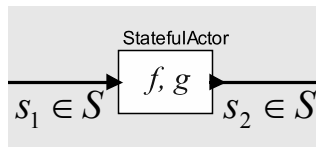
state space

$$\forall (t, n) \in T \times N, s_2(t, n) = ?$$

The new function f gives outputs in terms of inputs and the current state. The function g updates the state at the specified time.

Lee, Hybrid: 21

Abstract Semantics



$$S = [T \times N \rightarrow R]$$

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$

At each $t \in T$ the output is a *sequence* of one or more values where given the current state $\sigma(t) \in \Sigma$ and the input $s_1(t)$ we evaluate the procedure

$$s_2(t, 0) = f(\sigma(t), s_1(t), t)$$

$$\sigma_1(t) = g(\sigma(t), s_1(t), t)$$

$$s_2(t, 1) = f(\sigma_1(t), s_1(t), t)$$

$$\sigma_2(t) = g(\sigma_1(t), s_1(t), t)$$

...

until the state no longer changes. We use the final state on any evaluation at later times.

This deals with the first requirement.

Lee, Hybrid: 22

Generalizing: Multiple Events at the Same Time using Transient States



CT Director

- level1: 0.5
- level2: 1.25
- level3: -0.45

This model shows that the level crossing detectors detect both the continuities and discontinuities properly.

The three level crossing detectors detect levels 0.5, 1.25, and -0.45 respectively.

The modal model produces a piecewise continuous signal with glitches (produced by the output actions of the transient states.)

This finite state machine generates a piecewise-continuous signal with glitches.

The "init" state produces a consistent continuous signal. The states: state1, state2, and state3, are transient states. Their transitions produce glitches with their output actions.

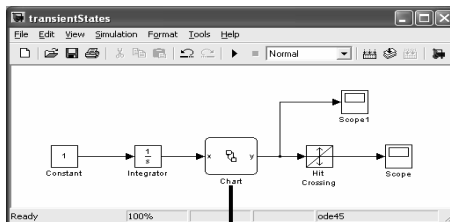
If an outgoing guard is true upon entering a state, then the time spent in that state is identically zero. This is called a "transient state."

Lee, Hybrid: 23

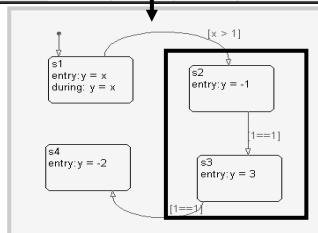
Contrast with Simulink/Stateflow



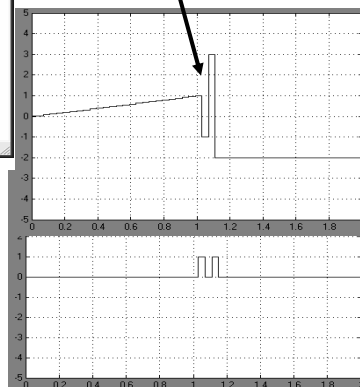
In Simulink semantics, a signal can only have one value at a given time. Consequently, Simulink introduces solver-dependent behavior.



The simulator engine of Simulink introduces a non-zero delay to consecutive transitions.



Transient States

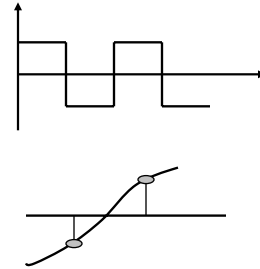


Second Requirement: Simulation Times Must Include Event Times



Event times are sometimes predictable (e.g. the times of discontinuous outputs of a clock) and sometimes unpredictable without running the solver (e.g. the time at which a continuous-time crosses a threshold). In both cases, the solver must not step over the event time.

- Predictable Breakpoints:
 - Known beforehand.
 - Register to a Breakpoint Table in advance.
 - Use breakpoints to adjust step sizes.
- Unpredictable Breakpoints:
 - Known only after they have been missed.
 - Requires being able to backtrack and re-execute with a smaller step size.

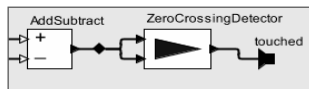


Lee, Hybrid: 25

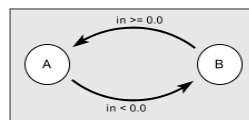
Event Times



In continuous-time models, Ptolemy II can use *event detectors* to identify the precise time at which an event occurs:



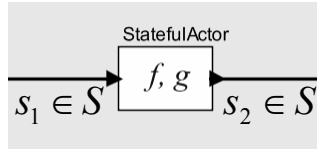
or it can use Modal Models, where guards on the transitions specify when events occur. In the literature, you can find two semantic interpretations to guards: *enabling* or *triggering*.



If only enabling semantics are provided, then it becomes nearly impossible to give models whose behavior does not depend on the step-size choices of the solver.

Lee, Hybrid: 26

The Abstract Semantics Supports the Second Requirement as Well



$$S = [T \times N \rightarrow R]$$

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$

At each $t \in T$ the calculation of the output given the input is separated from the calculation of the new state. Thus, the state does not need to be updated until after the step size has been decided upon.

In fact, the variable step size solver relies on this, since any of several integration calculations may result in refinement of the step size because the error is too large.

This deals with the second requirement.

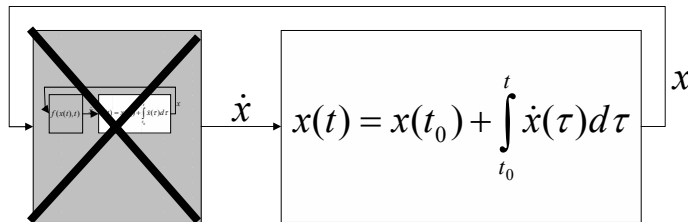
However, We Don't Quite Get Compositional Semantics



In general, the behavior of the inside solver cannot be given by functions of form:

$$f : \Sigma \times R^m \times T \rightarrow R^m$$

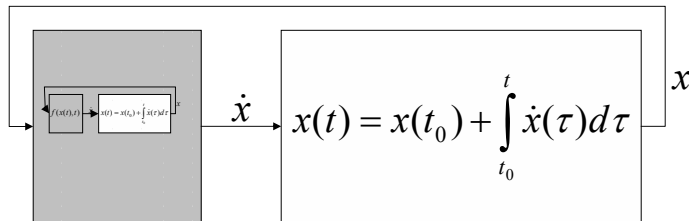
$$g : \Sigma \times R^m \times T \rightarrow \Sigma$$



Third Requirement: Compositional Semantics



We require that the system below yield an execution that is identical to a flattened version of the same system. That is, despite having two solvers, it must behave as if it had one.



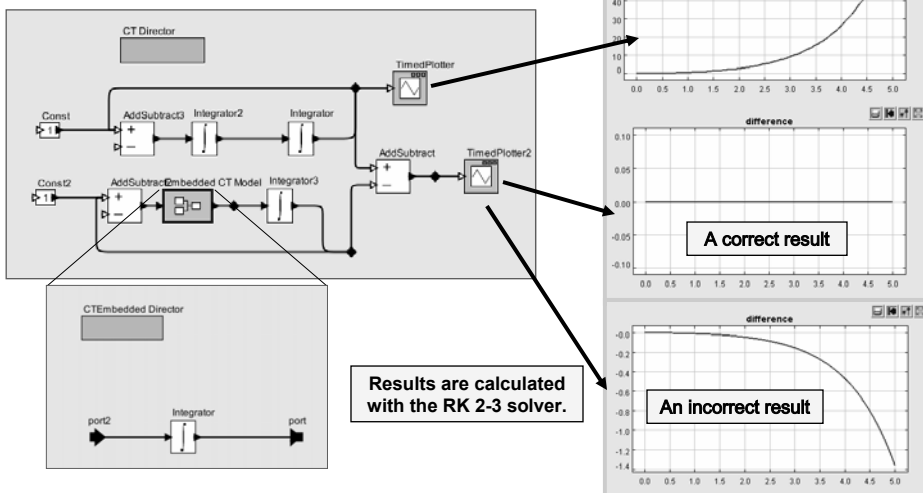
Achieving this appears to require that the two solvers coordinate quite closely. This is challenging when the hierarchy is deeper.

Lee, Hybrid: 29

Compositional Execution



Haiyang Zheng noticed that earlier versions of HyVisual did not exhibit compositional behavior.

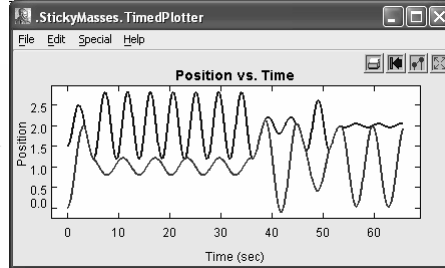


Lee, Hybrid: 30

The "Right Design" Supports Deeper Hierarchies

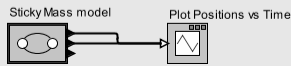


Masses on Springs



Continuous Time (CT) Director

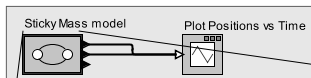
This model shows a hybrid system, which mixes continuous-time modeling with finite state machines. In this example, two point masses on springs oscillate. However, they may collide, in which case, they stick together, and oscillate together. The stickiness decays, and they eventually come apart again. This is an example of a modal model, where there are two modes, "together" and "separate". Each mode is modeled by a state in an FSM, and each state refines to a continuous-time model of the dynamics in that mode.



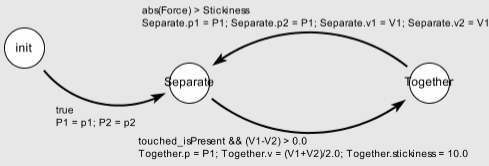
Consider two masses on springs which, when they collide, will stick together with a decaying stickiness until the force of the springs pulls them apart again.

Lee, Hybrid: 31

Structure of the Spring-Masses Model



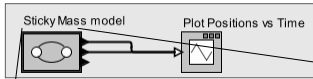
The sticky masses system has two modes of operation, "Separate" and "Together," corresponding to whether the point masses are stuck together. The "init" state has a transition that is used to initialize the "Separate" model (double click on that transition to see its actions).



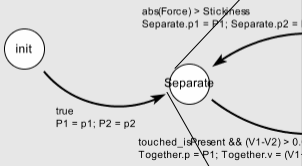
A component in a continuous-time model is defined by a finite state machine.

Lee, Hybrid: 32

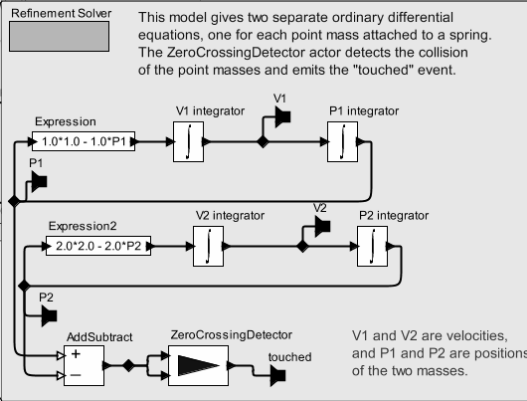
Structure of the Spring-Masses Model



The sticky masses system has two modes of operation, "Separate" and "Together," corresponding to whether the point masses are stuck together. The "init" state has a transition that is used to initialize the "Separate" model (double click on that transition to see its actions).



Each state has a "refinement," which is a contained model defining behavior.

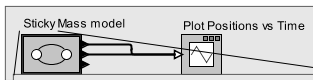


This model gives two separate ordinary differential equations, one for each point mass attached to a spring. The ZeroCrossingDetector actor detects the collision of the point masses and emits the "touched" event.

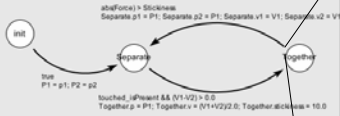
V1 and V2 are velocities, and P1 and P2 are positions of the two masses.

Notice that we need compositionality.

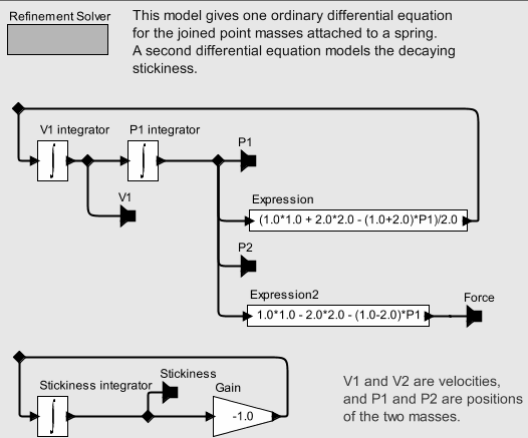
Structure of the Spring-Masses Model



The sticky masses system has two modes of operation, "Separate" and "Together," corresponding to whether the point masses are stuck together. The "init" state has a transition that is used to initialize the "Separate" model (double click on that transition to see its actions).



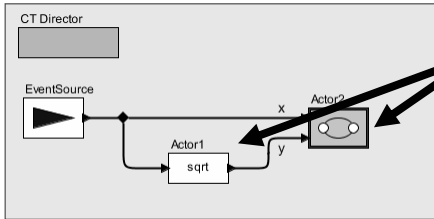
State refinements are inactive when the FSM is not in that state. An arc into a state can specify a reset map, or it can resume the refinement in the state where it last left off.



This model gives one ordinary differential equation for the joined point masses attached to a spring. A second differential equation models the decaying stickiness.

V1 and V2 are velocities, and P1 and P2 are positions of the two masses.

Simultaneous Events: The Order of Execution Question



Given an event from the event source, which of these should react first? Nondeterministic? Data precedences?

Simulink/Stateflow and HyVisual declare this to be deterministic, based on data precedences. Actor1 executes before Actor2.

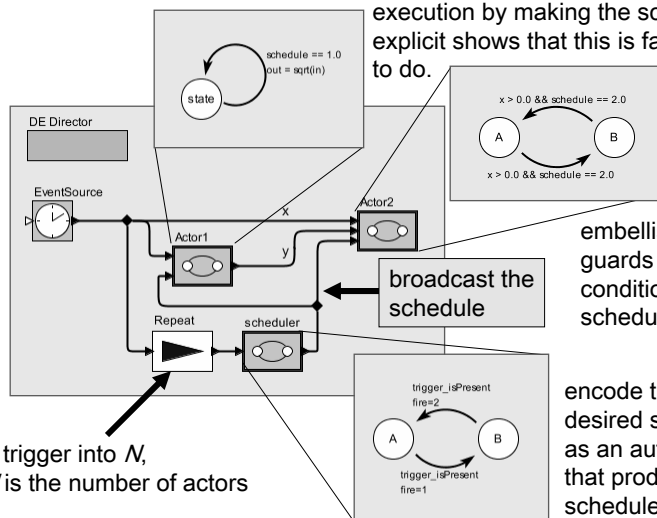
Many formal hybrid systems languages (e.g. Charon) declare this to be nondeterministic.

Semantics of a signal:

$$s : T \times N \rightarrow R$$

In HyVisual, every continuous-time signal has a value at $(t, 0)$ for any $t \in T$. This yields deterministic execution of the above model.

Non-Deterministic Interaction is the Wrong Answer



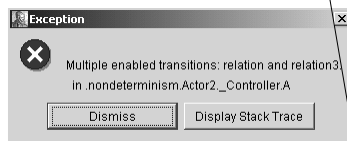
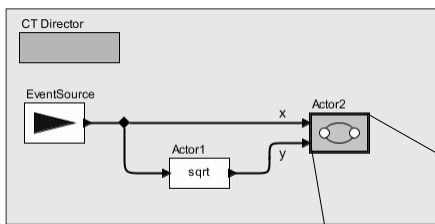
An attempt to achieve deterministic execution by making the scheduling explicit shows that this is far too difficult to do.

embellish the guards with conditions on the schedule

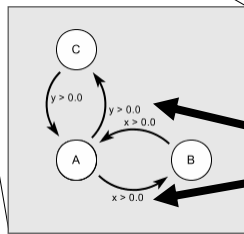
encode the desired sequence as an automaton that produces a schedule

turn one trigger into N , where N is the number of actors

OTOH: Nondeterminism is Easily Added in a Deterministic Modeling Framework



Although this can be done in principle, HyVisual does not support this sort of nondeterminism. What execution trace should it give?



At a time when the event source yields a positive number, both transitions are enabled.

Lee, Hybrid: 37

Nondeterministic Ordering



In favor

- Physical systems have no true simultaneity
- Simultaneity in a model is artifact
- Nondeterminism reflects this physical reality

Against

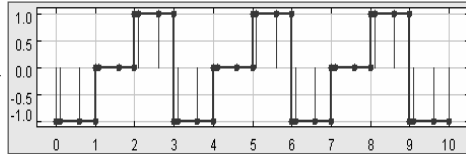
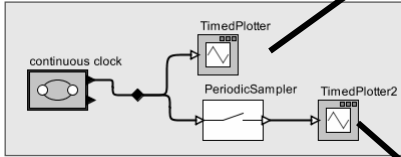
- It surprises the designer
 - counters intuition about causality
- It is hard to get determinism
 - determinism is often desired (to get repeatability)
- Getting the desired nondeterminism is easy
 - build on deterministic ordering with nondeterministic FSMs
- Writing simulators that are trustworthy is difficult
 - It is incorrect to just pick one possible behavior!

Lee, Hybrid: 38

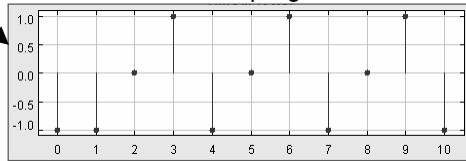
Sampling Discontinuous Signals



Continuous signal with sample times chosen by the solver:



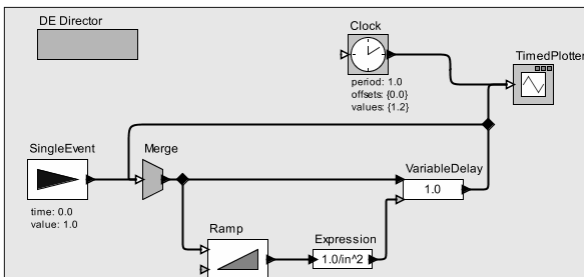
Discrete result of sampling:



Samples must be deterministically taken at t^- or t^+ . Our choice is t^- , inspired by hardware setup times.

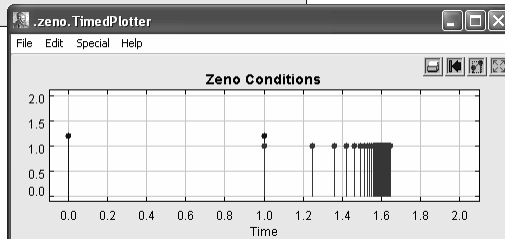
Note that in HyVisual, unlike Simulink, discrete signals have no value except at discrete points.

Zeno Conditions



This model illustrates a Zeno condition, where an infinite number of events occur before time 2.0, and hence the Clock actor is unable to ever produce its output at time 2.0.

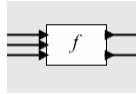
Zeno behavior is a property of the discrete events in a system, not a property of its continuous dynamics. The continuous dynamics merely determine the time between events.



Consequently, Zeno Behavior Can Be Dealt With (almost) Entirely in Discrete Events.



Let the set of all signals be $S = [T \times N \rightarrow V]$ where V is a set of values. Let an actor



be a function $F : S^n \rightarrow S^m$. What are the constraints on such functions such that:

1. Compositions of actors are determinate.
2. Feedback compositions have a meaning.
3. We can rule out Zeno behavior.

Lee, Hybrid: 41

The Cantor Metric



Given the tag set $T \times N$ use only the time stamps. Let

$$d : [T \times N \rightarrow V] \times [T \times N \rightarrow V] \rightarrow R$$

such that for all $s, s' \in [T \times N \rightarrow V]$,

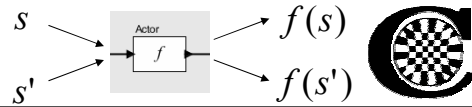
$$d(s, s') = 1/2^\tau$$

where τ is the greatest lower bound of $\{t \in T \mid s(t) \neq s'(t)\}$.

The function d is an ultrametric.

Lee, Hybrid: 42

Causality



Causal. For all signals s and s'

$$d(f(s), f(s')) \leq d(s, s')$$

Strictly causal. For all signals s and s'

$$s \neq s' \Rightarrow d(f(s), f(s')) < d(s, s')$$

Delta causal. There exists a real number $\delta < 1$ such that for all signals s and s'

$$s \neq s' \Rightarrow d(f(s), f(s')) \leq \delta d(s, s')$$

Lee, Hybrid: 43

Fixed Point Theorem



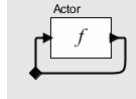
Let $(S^n = [T \times N \rightarrow V]^n, d)$ be a metric space and $f: S^n \rightarrow S^n$ be a strictly causal function. Then f has at most one fixed point.

This takes care of determinacy. There can be no more than one behavior.

Can we find that behavior?

Lee, Hybrid: 44

Fixed Point Theorem 4 (Banach Fixed Point Theorem)



Let $(S^n = [T \times N \rightarrow V]^n, d)$ be a *complete* metric space and $f: S^n \rightarrow S^n$ be a delta causal function. Then f has a unique fixed point, and for any point $s \in S^n$, the following sequence converges to that fixed point:

$$s_1 = s, s_2 = f(s_1), s_3 = f(s_2), \dots$$

This means no Zeno! Notes:

- Our metric space is complete (take my class)
- Convergence means that time advances to infinity!

Lee, Hybrid: 45

Observations

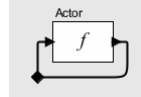


If there is a lower bound on the step size:

- All signals are discrete
 - there is an order embedding to the natural numbers
- Integrators with the RK2-3 solver are delta causal, so
 - solution with feedback is unique
 - no Zeno in discretized steps
 - but lower bound on the step size implies inaccuracies
- Integrators with some methods (e.g. trapezoidal rule) are not delta causal, nor even strictly causal, so we have no assurance of a unique solution in feedback systems.

Lee, Hybrid: 46

Extensions



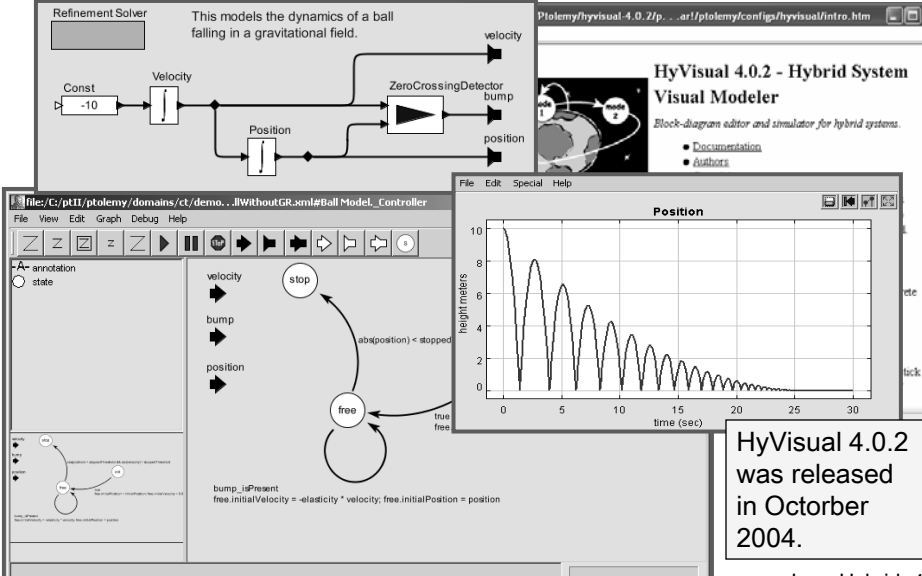
The Cantor metric formulation ignores the index part of the tag in the signal space $S^n = [T \times N \rightarrow V]^n$. Consider for example an actor in a feedback loop that models software by reacting to a finite number n of events in zero time, and then introduces a delay in its reaction to the $n + 1$ event. This actor is not even strictly causal under the classical formulation.

We have two extensions that tighten the model:

- Generalized ultrametric space where the metric is not a real number, but a value in a well-founded total order [Cataldo, Lee, Liu, Matiskoudis]
- Adaptation of the Cantor metric that leverages the structure of discrete-event signals to account for the index [Lee and Zheng].

Lee, Hybrid: 47

Conclusion: HyVisual – Executable Hybrid System Modeling Built on Ptolemy II



The screenshot displays the HyVisual 4.0.2 interface. At the top, a block diagram models a ball falling in a gravitational field. It includes a 'Const' block with the value -10, a 'Velocity' block, a 'Position' block, and a 'ZeroCrossingDetector' block. The detector outputs 'bump' and 'position' signals. Below the block diagram is a state transition diagram with states 'stop' and 'free'. Transitions are labeled with conditions like 'ab(position) < stopped' and 'true free'. A graph titled 'Position' shows 'Height meters' on the y-axis (0 to 10) and 'time (sec)' on the x-axis (0 to 30). The graph shows a series of peaks and troughs that decay over time, representing the ball's oscillatory motion. A text box in the bottom right corner states: 'HyVisual 4.0.2 was released in October 2004.'

Lee, Hybrid: 48