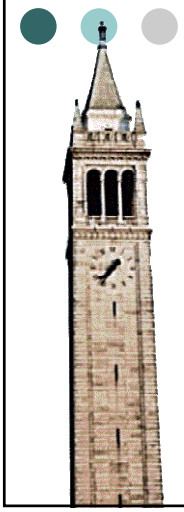


The Operational Semantics of Hybrid Systems

Edward A. Lee
Professor, Chair of EE, and
Associate Chair of EECS,
UC Berkeley

With contributions from:
Adam Cataldo, Jie Liu, Xiaojun Liu,
Eleftherios Matsikoudis, and Haiyang Zheng

Invited Plenary Talk
Hybrid Systems: Computation and Control (HSCC)
Zurich, Switzerland, March 9, 2005

Chess
Center for Hybrid and
Embedded Software Systems

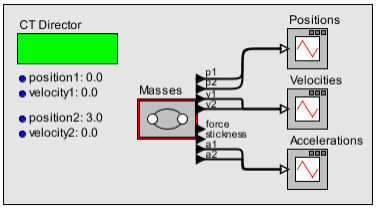
The Premise

Hybrid Systems can be thought of as executable programs. In this case, they need to be given an executable semantics.

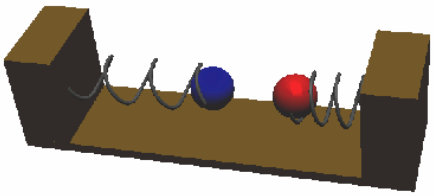
Outline

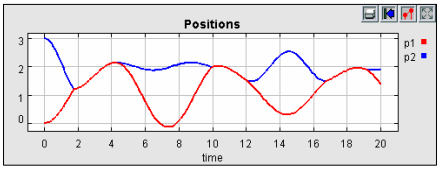
- Signals with discontinuities
- Ideal solver semantics
- Choosing step sizes
- Discrete phase of execution
- Miscellaneous issues
  - Enabling vs. triggering guards
  - Order of reactions to simultaneous events
  - Nondeterministic state machines
  - Sampling discontinuous signals
  - Zeno behaviors

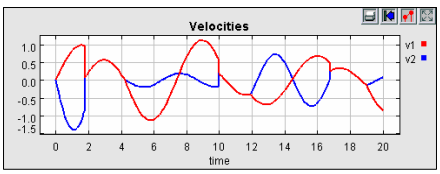
A Hybrid Systems Example

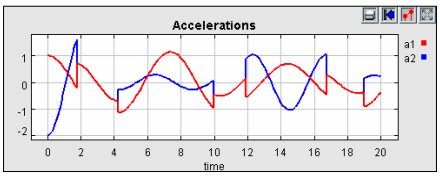


Consider two masses on springs which, when they collide, will stick together with a decaying stickiness until the force of the springs pulls them apart again.

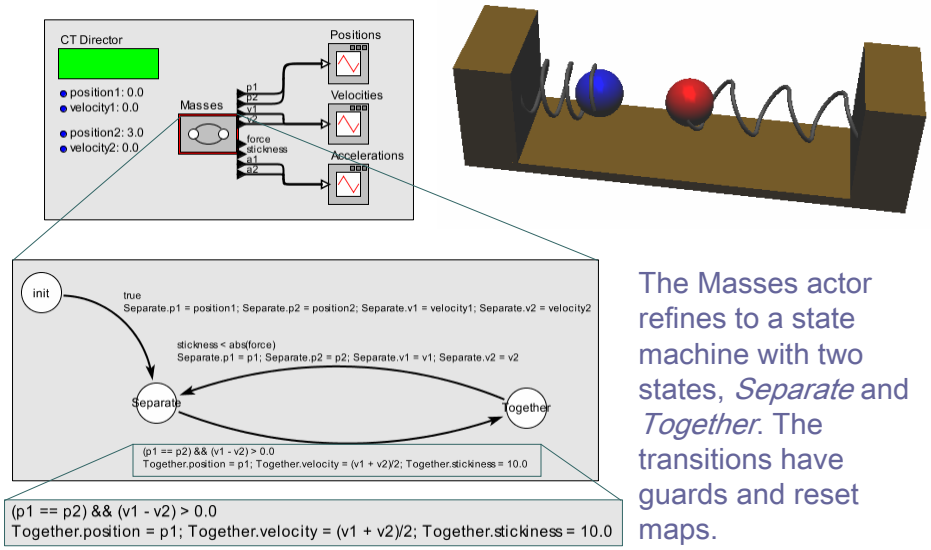






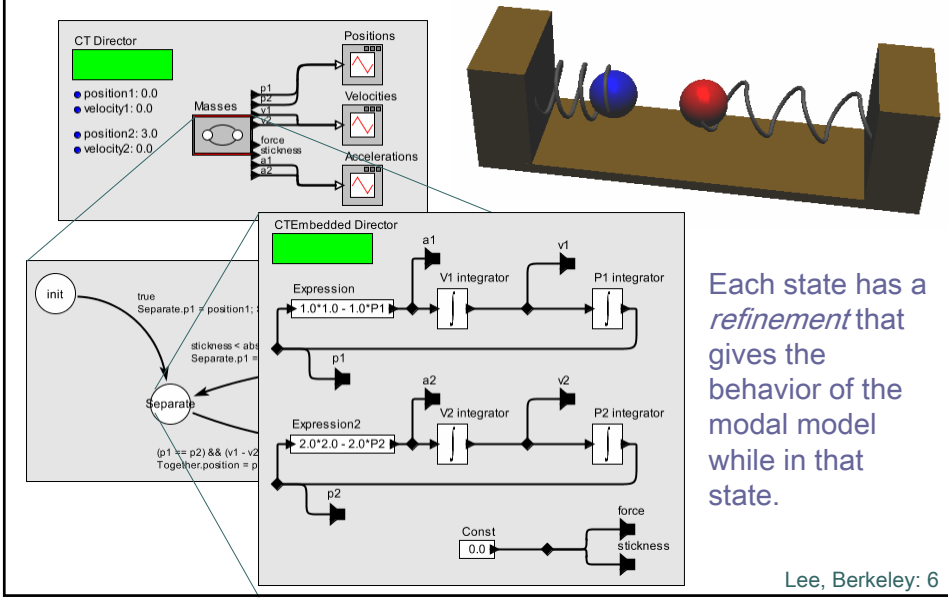


Modal Models



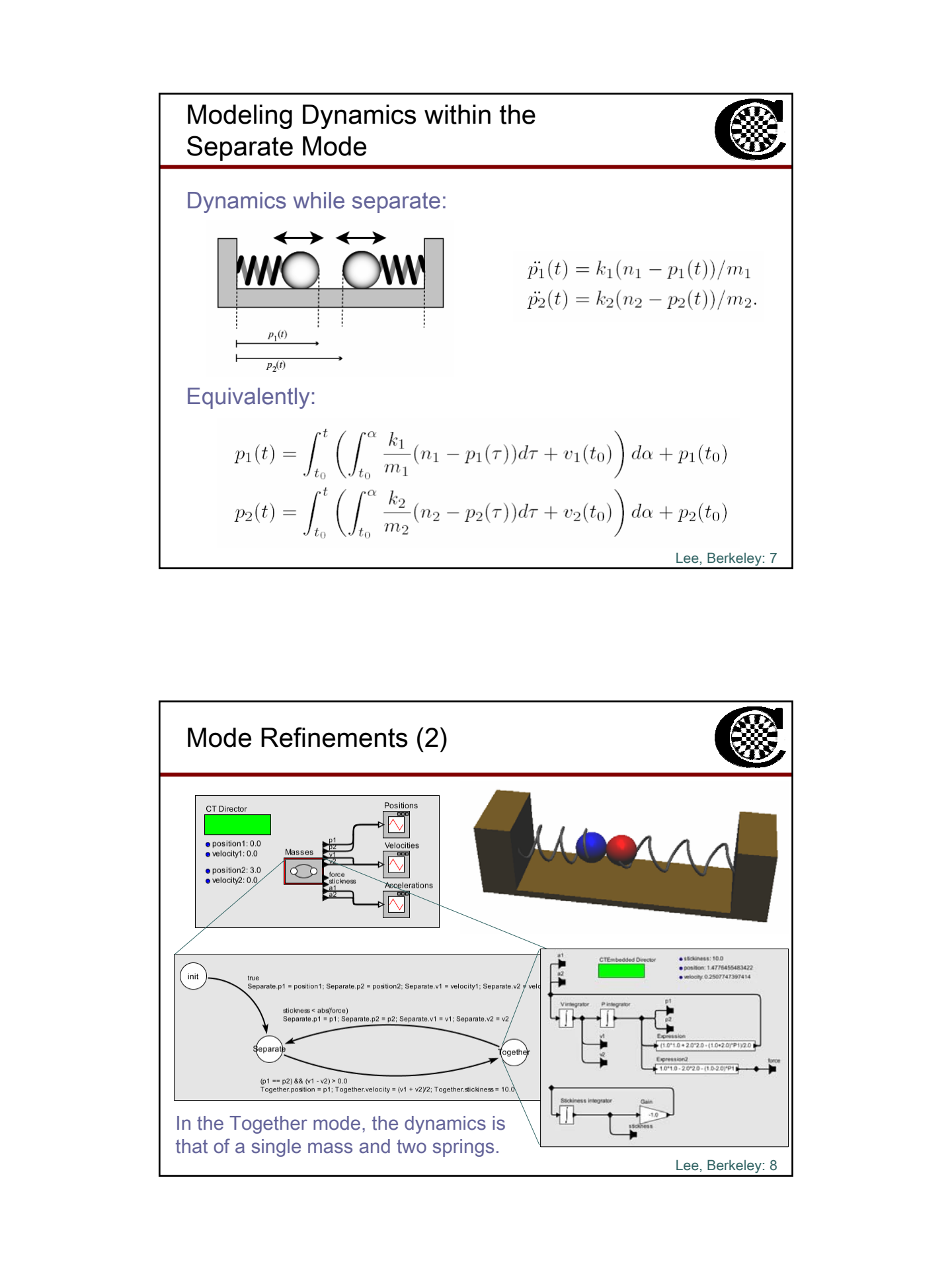
The Masses actor refines to a state machine with two states, *Separate* and *Together*. The transitions have guards and reset maps.

Mode Refinements



Each state has a *refinement* that gives the behavior of the modal model while in that state.

Modeling Dynamics within the Separate Mode



Dynamics while separate:

$$\ddot{p_1}(t) = k_1(n_1 - p_1(t))/m_1$$
$$\ddot{p_2}(t) = k_2(n_2 - p_2(t))/m_2.$$

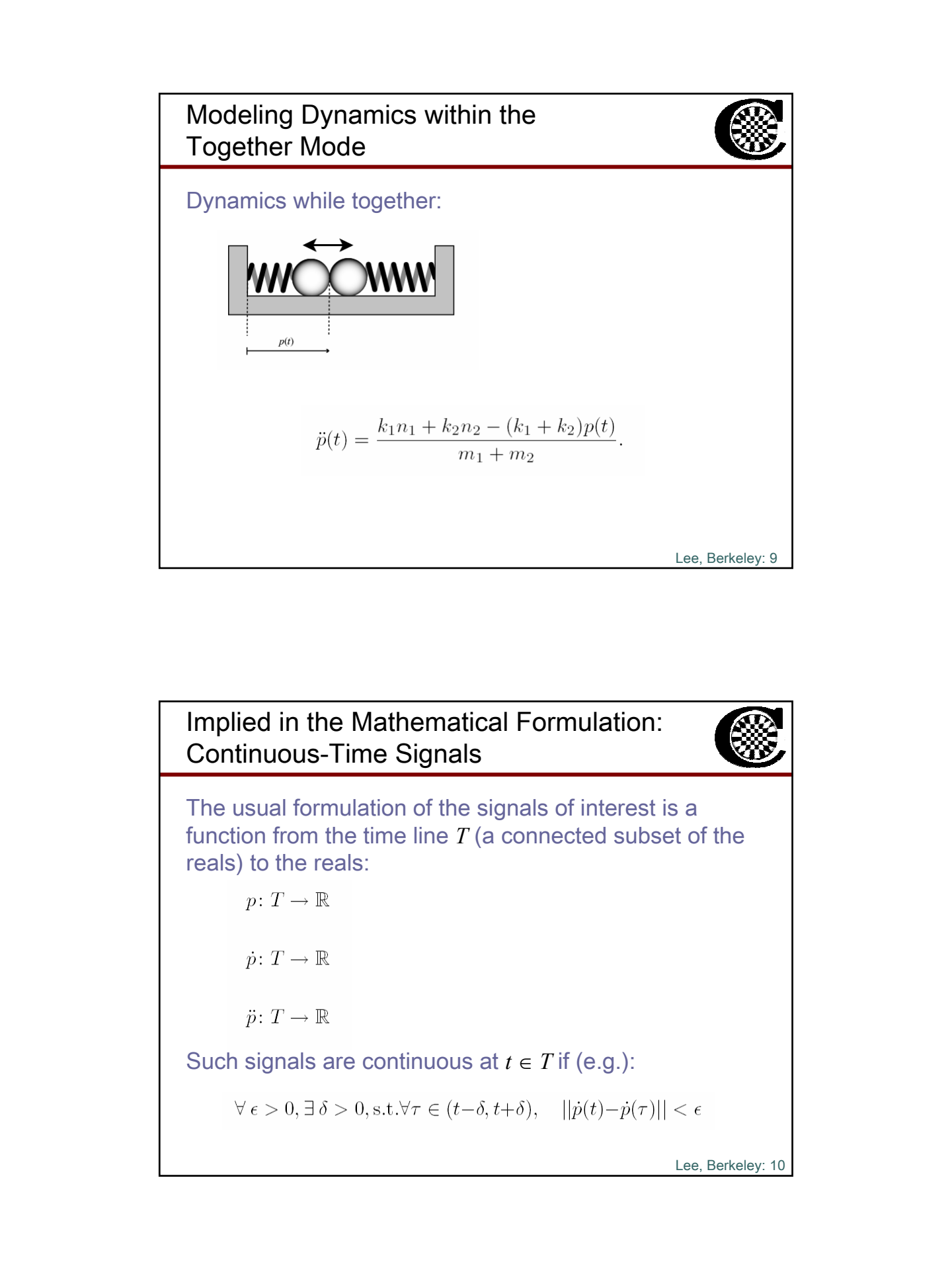
Equivalently:

$$p_1(t) = \int_{t_0}^{t} \left(\int_{t_0}^{\alpha} \frac{k_1}{m_1}(n_1 - p_1(\tau))d\tau + v_1(t_0) \right) d\alpha + p_1(t_0)$$
$$p_2(t) = \int_{t_0}^{t} \left(\int_{t_0}^{\alpha} \frac{k_2}{m_2}(n_2 - p_2(\tau))d\tau + v_2(t_0) \right) d\alpha + p_2(t_0)$$

Mode Refinements (2)

In the Together mode, the dynamics is that of a single mass and two springs.

Modeling Dynamics within the Together Mode



Dynamics while together:

$$\ddot{p}(t) = \frac{k_1 n_1 + k_2 n_2 - (k_1 + k_2)p(t)}{m_1 + m_2}.$$

Implied in the Mathematical Formulation: Continuous-Time Signals

The usual formulation of the signals of interest is a function from the time line T (a connected subset of the reals) to the reals:

$$p\colon T \to \mathbb{R}$$

$$\dot{p}\colon T \to \mathbb{R}$$

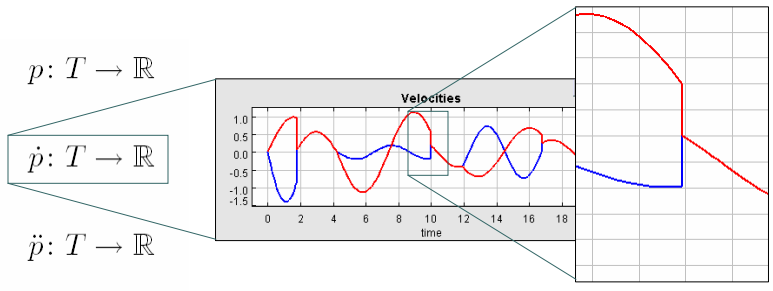
$$\ddot{p}\colon T \to \mathbb{R}$$

Such signals are continuous at $t \in T$ if (e.g.):

$$\forall\, \epsilon > 0, \exists\, \delta > 0, \text{s.t.} \forall \tau \in (t-\delta, t+\delta), \quad ||\dot{p}(t) - \dot{p}(\tau)|| < \epsilon$$

Piecewise Continuous Signals

In hybrid systems of interest, signals have discontinuities.

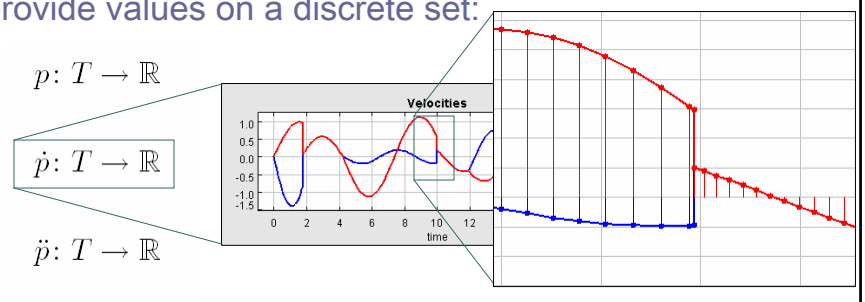


Piecewise continuous signals are continuous at all $t \in T \setminus D$ where $D \subset T$ is a *discrete set*.[1]

[1]A set D with an order relation is a *discrete set* if there exists an order embedding to the integers).

Operational Semantics of Hybrid Systems

A computer execution of a hybrid system is constrained to provide values on a discrete set:



Given this constraint, choosing $T \subset \mathbb{R}$ as the domain of these functions is an unfortunate choice. It makes it impossible to unambiguously represent discontinuities.

Definition: *Continuously Evolving Signal*

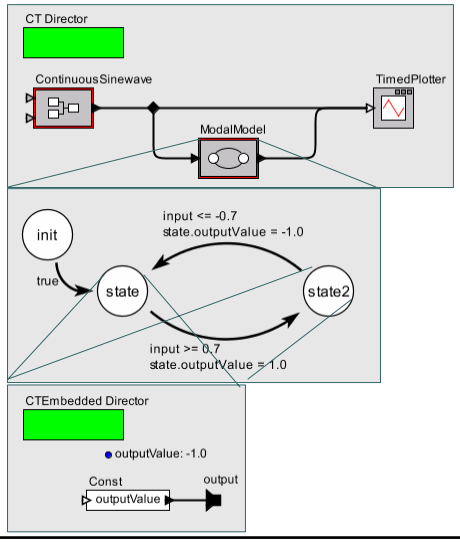
Change the domain of the function:

$$x : T \times \mathbb{N} \rightarrow V$$

Where T is a connected subset of the reals and $\mathbb{N}$ is the set of natural numbers.

At each time $t \in T$, the signal x has a sequence of values. Where the signal is continuous, all the values are the same. Where is discontinuous, it has multiple values.

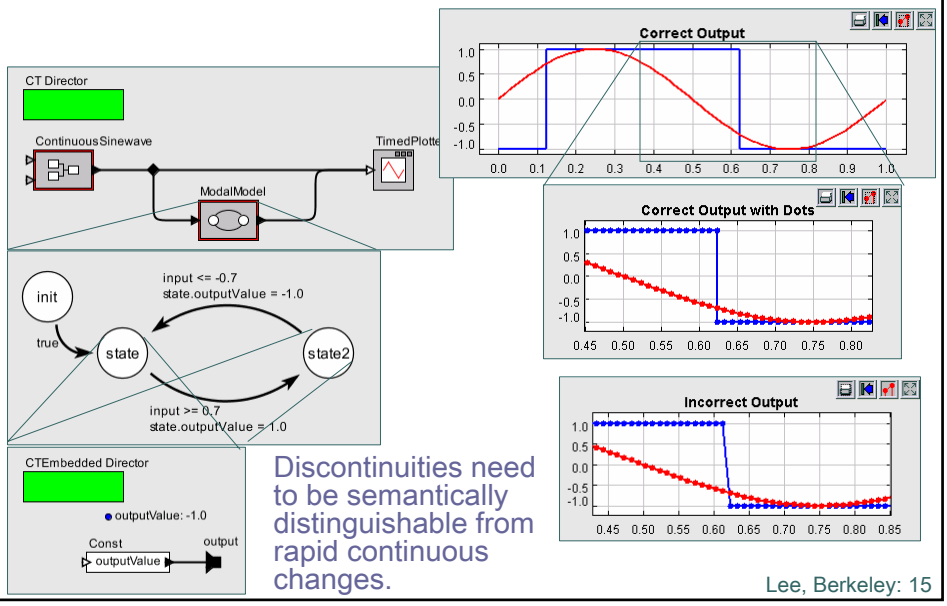
Simpler Example: Hysteresis



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*.

Signals Must Have Multiple Values at the Time of a Discontinuity



Discontinuities need to be semantically distinguishable from rapid continuous changes.

Initial and Final Value Signals

A signal $x : T \times \mathbb{N} \rightarrow V$ has no *chattering Zeno* condition if there is an integer $m > 0$ such that

$$\forall n > m, \quad x(t,n) = x(t,m)$$

A non-chattering signal has a corresponding *final value signal*, $x_f : T \rightarrow V$ where

$$\forall\ t \in T, \quad x_f(t) = x(t,m)$$

It also has an *initial value signal* $x_i : T \rightarrow V$ where

$$\forall\ t \in T, \quad x_i(t) = x(t,0)$$

Piecewise Continuous Signals

A piecewise continuous signal is a non-chattering signal

$$x : T \times \mathbb{N} \rightarrow V$$

where

- The initial signal x_i is continuous on the left,
- The final signal x_f is continuous on the right, and
- The signal x has only one value at all $t \in T \setminus D$ where $D \subset T$ is a discrete set.

Outline

- Signals with discontinuities
- Ideal solver semantics
- Choosing step sizes
- Discrete phase of execution
- Miscellaneous issues
  - Enabling vs. triggering guards
  - Order of reactions to simultaneous events
  - Nondeterministic state machines
  - Sampling discontinuous signals
  - Zeno behaviors

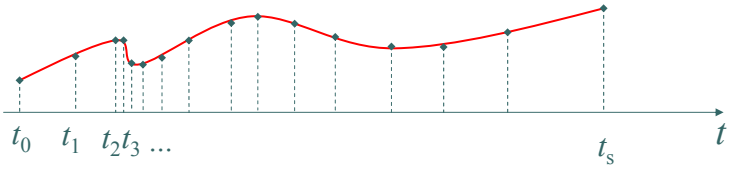
Discrete Trace: What it Means to *Execute* a Hybrid System

Let $D' \subset T$ be a discrete set that includes at least the initial time and the times of all discontinuities.

A *discrete trace* of the signal x is a set:

$$\{x(t,n) \mid t \in D', \text{ and } n \in \mathbb{N}\}$$

An execution of a hybrid system is the construction of a discrete trace:

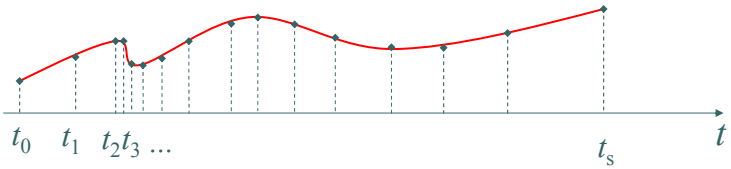


Ideal Solver Semantics

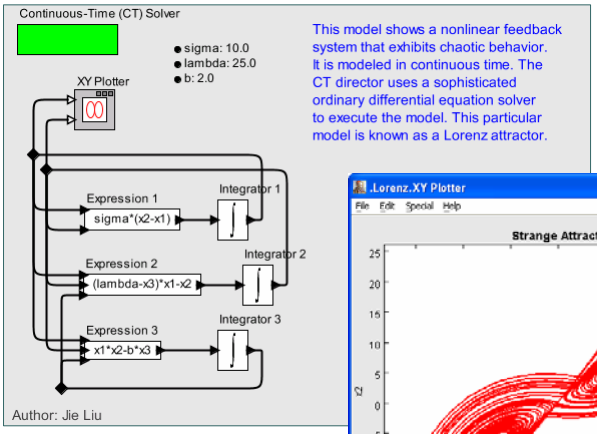
[Liu and Lee, HSCC 2003]

In the *ideal solver semantics,* the ODE governing the hybrid system has a unique solution for all intervals $[t_i\,,\, t_{i+1})$ for each neighboring $t_i < t_{i+1} \in D'$. The discrete trace loses nothing by not representing values within these intervals.

Although an idealization, this is not far fetched. The spring masses example, for instance, conforms with the assumptions and can be executed by an ideal solver.

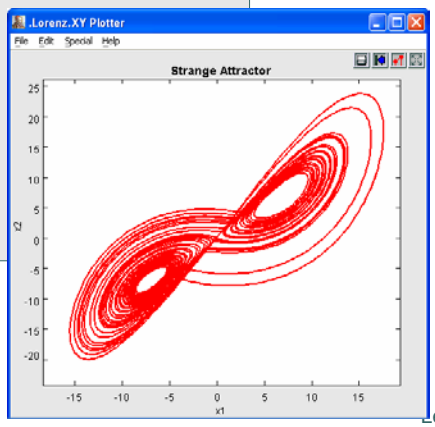


Modeling Continuous Dynamics with Discrete Traces

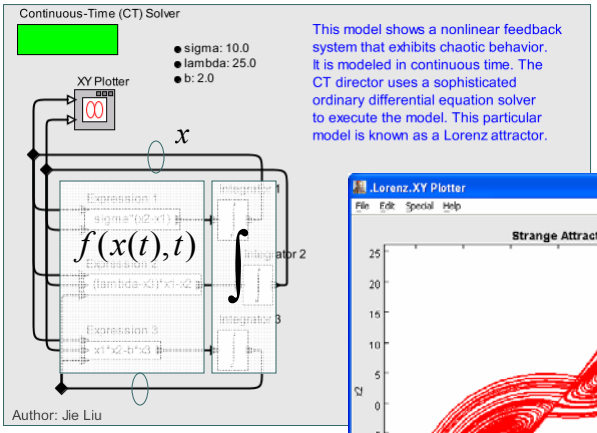


This model shows a nonlinear feedback system that exhibits chaotic behavior. It is modeled in continuous time. The CT director uses a sophisticated ordinary differential equation solver to execute the model. This particular model is known as a Lorenz attractor.

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

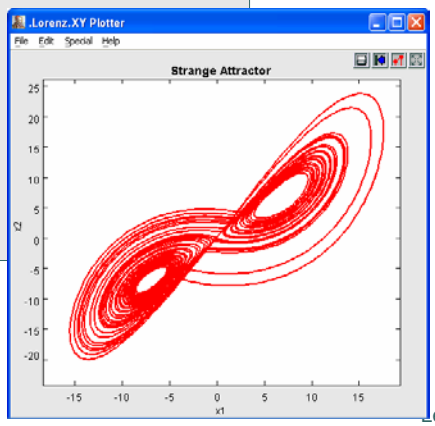


Structure of the Model of Continuous Dynamics



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.

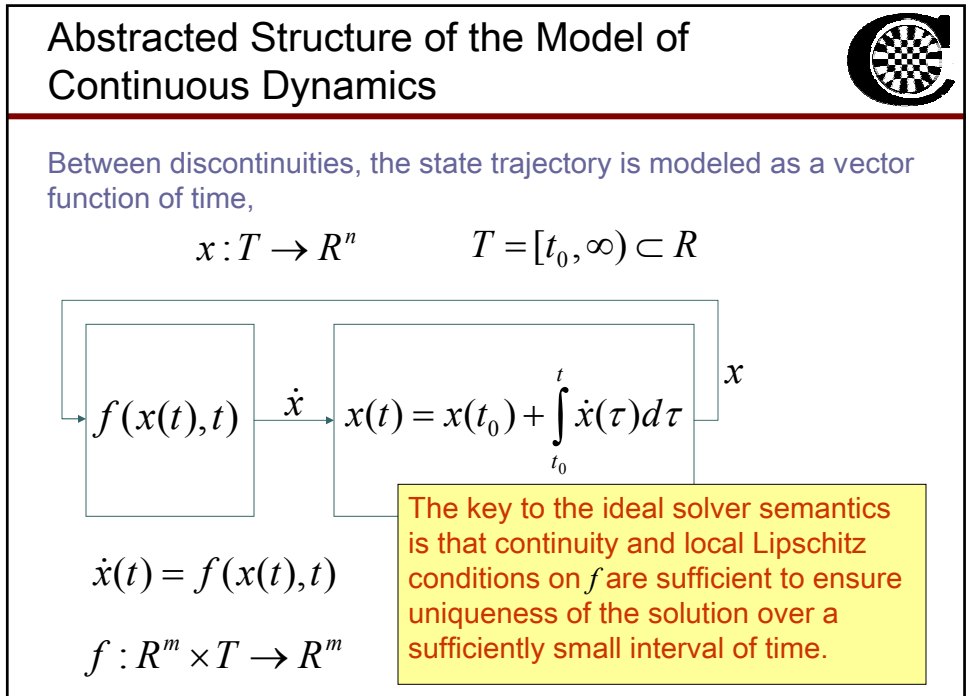
A basic continuous-time model describes an ordinary differential equation (ODE).



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

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

Abstracted Structure of the Model of Continuous Dynamics



Between discontinuities, the state trajectory is modeled as a vector function of time,

$$x : T \to R^n \qquad T = [t_0, \infty) \subset R$$

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

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

The key to the ideal solver semantics is that continuity and local Lipschitz conditions on f are sufficient to ensure uniqueness of the solution over a sufficiently small interval of time.

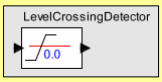
Outline

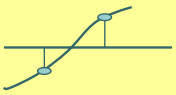
- Signals with discontinuities
- Ideal solver semantics
- Choosing step sizes
- Discrete phase of execution
- Miscellaneous issues
  - Enabling vs. triggering guards
  - Order of reactions to simultaneous events
  - Nondeterministic state machines
  - Sampling discontinuous signals
  - Zeno behaviors

Points on the Time Line that Must Be Included in a Discrete Trace

- Predictable breakpoints
  - Can be registered in advance with the solver

- Unpredictable breakpoints
  - Known after they have been missed





- Points that make the step size "sufficiently small"
  - Dependent on error estimation in the solver

Require backtracking

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

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

$$K_0 = f(x(t_n), t_n) \quad \leftarrow \dot{x}(t_n)$$

$$K_1 = f(x(t_n) + 0.5hK_0, t_n + 0.5h) \quad \leftarrow \text{estimate of } \dot{x}(t_n + 0.5h)$$

$$K_2 = f(x(t_n) + 0.75hK_1, t_n + 0.75h) \quad \leftarrow \text{estimate of } \dot{x}(t_n + 0.75h)$$

then let

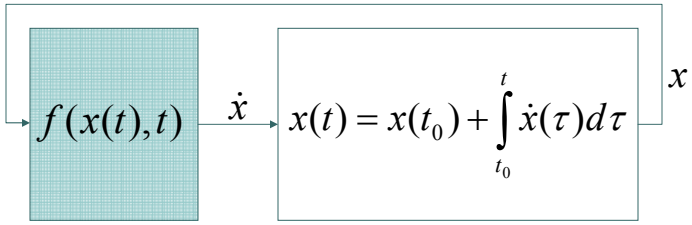
$$t_{n+1} = t_n + h$$

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

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

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).



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

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

Adjusting the Time Steps

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

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

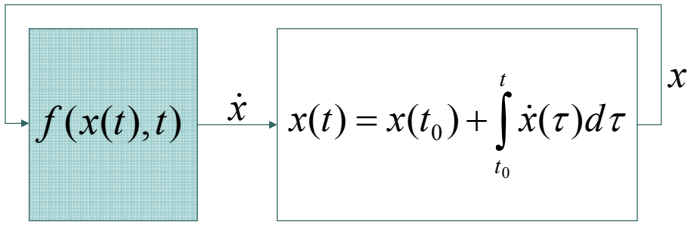
$$\varepsilon = h((-5/72)K_0 + (1/12)K_1 + (1/9)K_2 + (-1/8)K_3)$$

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.

Examining This Computationally

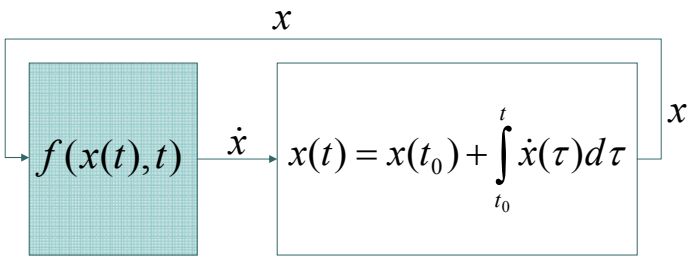


At each discrete time t_n, given a time increment $t_{n+1} = t_n + h$, we can estimate $x(t_{n+1})$ by repeatedly evaluating f with different values for the arguments. We may then decide that h is too large and reduce it and redo the process.

How General Is This Model?

Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines? no
- The model itself as a function? no

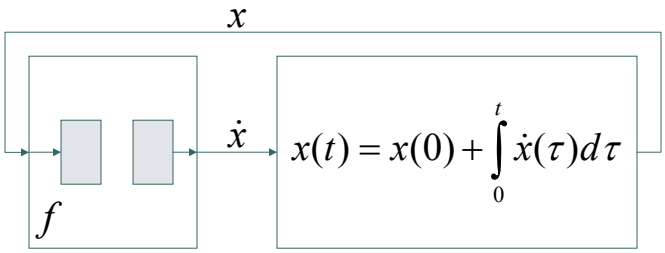


$$\dot{x}(t) = f(x(t),t)$$

How General Is This Model?

Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines? no
- The model itself as a function? no

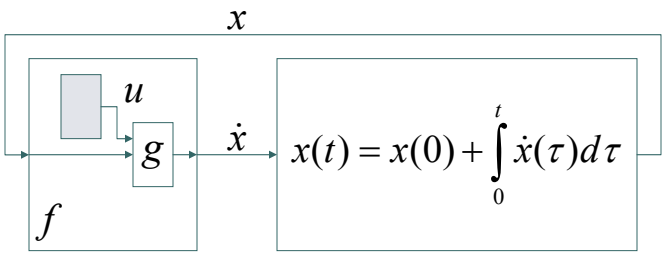


$$\dot{x}(t) = f(x(t), t)$$

How General Is This Model?

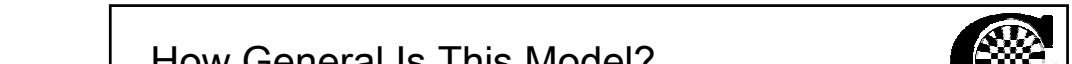
Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines? no
- The model itself as a function? no



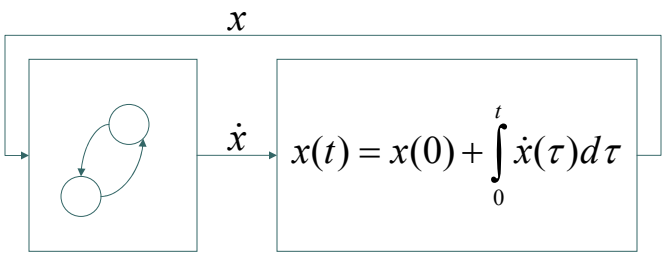
$$\dot{x}(t) = f(x(t), t) = g(u(t), x(t), t)$$

How General Is This Model?



Does it handle:

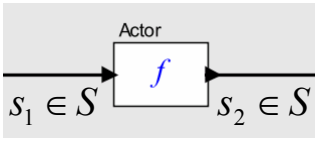
- Systems without feedback? yes
- External inputs? yes
- State machines? no, not immediately
- The model itself as a function? no



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

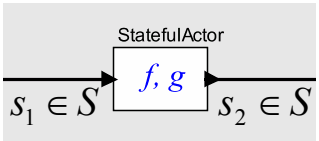
Actors with State Must Expose that State

Basic actor with firing:



$$S = [T \to R]$$
$$f : R^m \times T \to R^m$$
$$\forall t \in T, \quad s_2(t) = f(s_1(t), t)$$

Stateful actor:



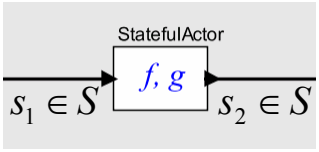
$$S = [T \times N \to R]$$
$$f : \Sigma \times R^m \times T \to R^m$$
$$g : \Sigma \times R^m \times T \to \Sigma$$

state space

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

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

Stateful Actors Support Unpredictable Breakpoints and Step Size Adaptation



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

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

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

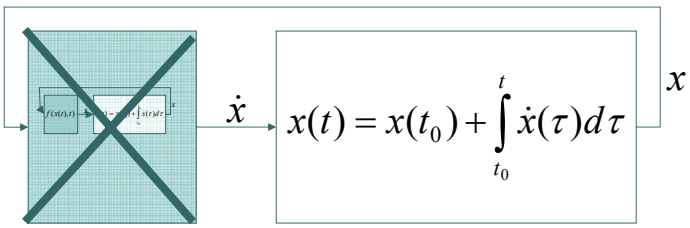
At each $t \in T$ the calculation of the output given the input is separated from the calculation of the new state. Thus, the state does not need to updated until after the step size has been decided upon.

In fact, a 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.

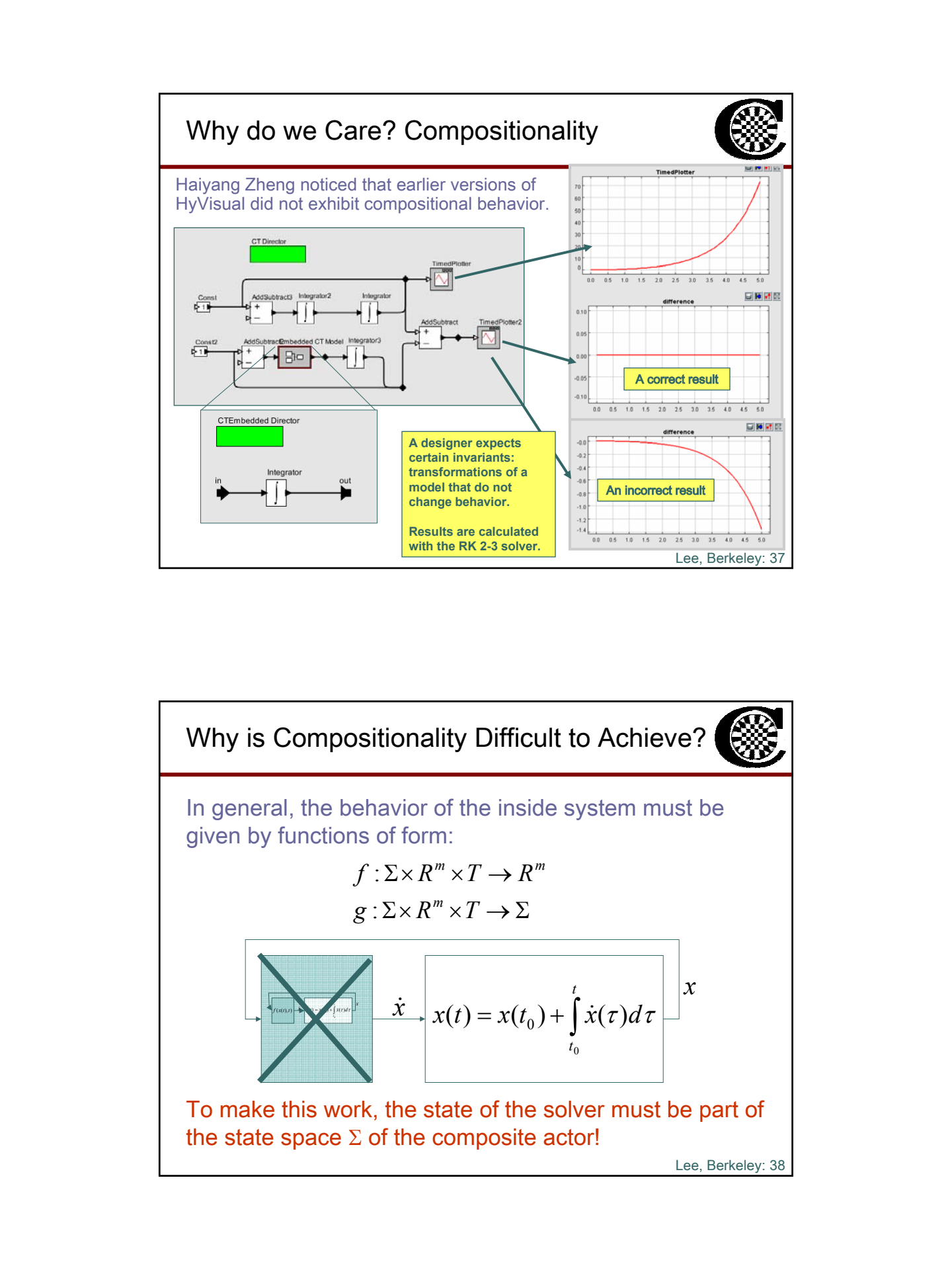
How General Is This Model?

Does it handle:

- Systems without feedback? yes
- External inputs? yes
- State machines? yes, with stateful actors
- The model itself as a function? yes, but be careful!



Why do we Care? Compositionality



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

A designer expects certain invariants: transformations of a model that do not change behavior.

Results are calculated with the RK 2-3 solver.

Why is Compositionality Difficult to Achieve?

In general, the behavior of the inside system must be given by functions of form:

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

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

To make this work, the state of the solver must be part of the state space Σ of the composite actor!

Compositional Execution Requires that Solvers Expose Details

An RK 2-3 solver evaluates signal values at intermediate points in time that do not truly qualify as a step. Given two RK 2-3 solvers in a hierarchy, if they do not cooperate on this, then the behavior is altered by the hierarchy.

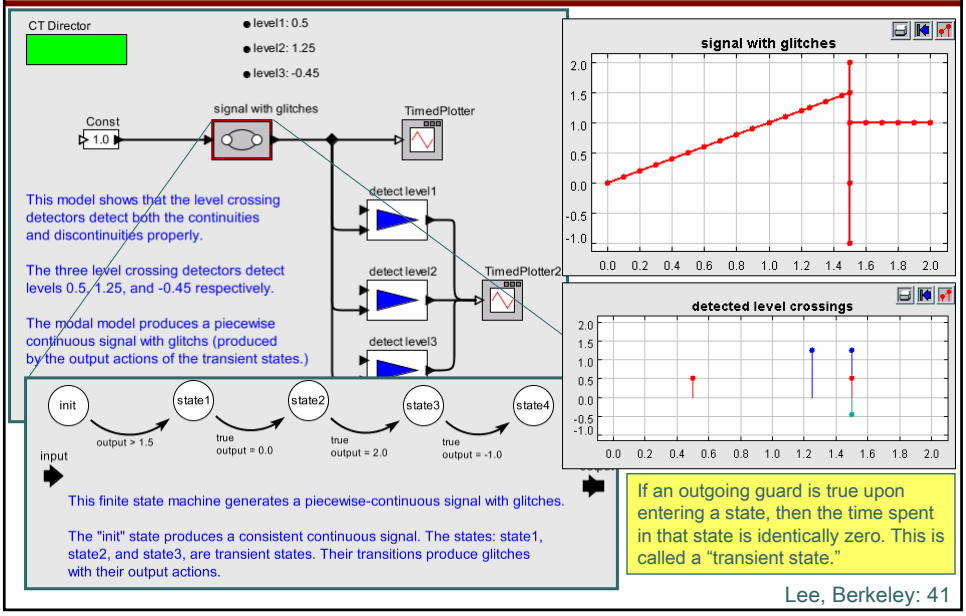
The HyVisual Solution: Solvers that are separated in the hierarchy by at most a Modal Model cooperate if they are the same type of solver.

- This is compositional, but
- This also allows heterogeneous mixtures of solvers.

Outline

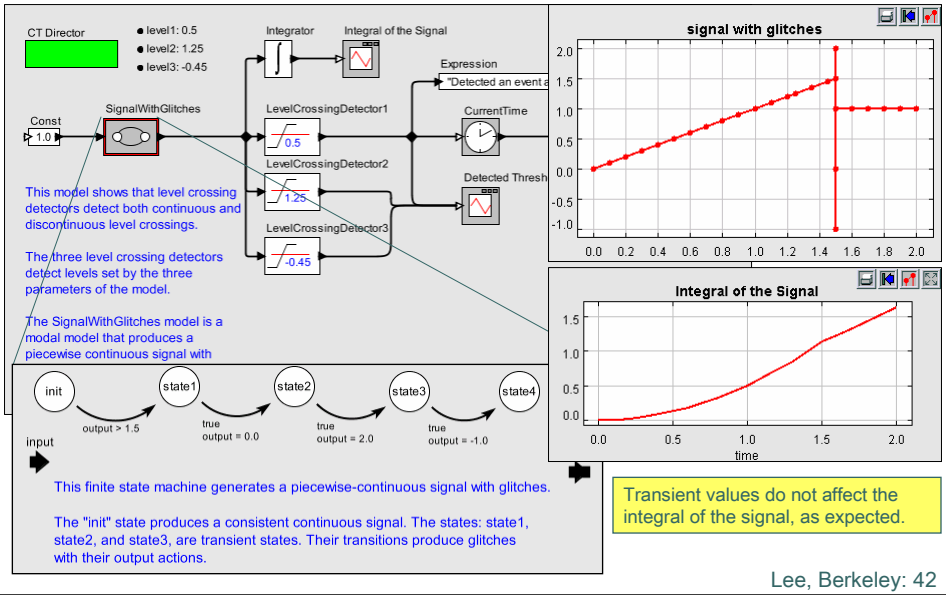
- Signals with discontinuities
- Ideal solver semantics
- Choosing step sizes
- Discrete phase of execution
- Miscellaneous issues
  - Enabling vs. triggering guards
  - Order of reactions to simultaneous events
  - Nondeterministic state machines
  - Sampling discontinuous signals
  - Zeno behaviors

Transient States
A Useful Model for Software



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.”

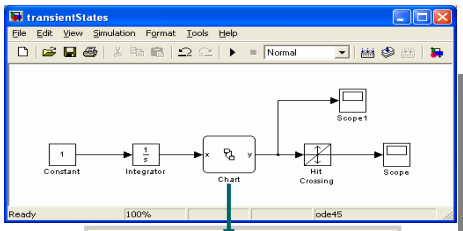
Transient Values Integrate to Zero

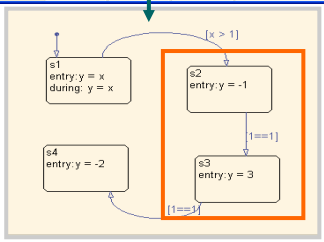


Transient values do not affect the integral of the signal, as expected.

Contrast with Simulink/Stateflow

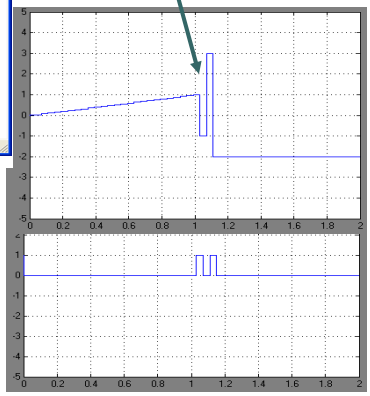
In Simulink semantics, a signal can only have one value at a given time. Consequently, Simulink introduces solver-dependent behavior.



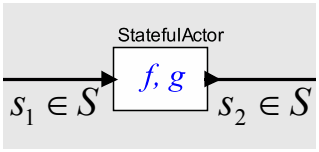


Transient States

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



Discrete Phase of Execution



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

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

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

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

$$s_2(t,0) = f(\sigma(t), s_1(t), t)$$
$$\sigma_1(t) = g(\sigma(t), s_1(t), t)$$ ← Commit to step here
$$s_2(t,1) = f(\sigma_1(t), s_1(t), t)$$
$$\sigma_2(t) = g(\sigma_1(t), s_1(t), t)$$
$$\ldots$$

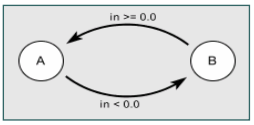
until the state no longer changes. We use the final state on any evaluation at later times.

Outline

- Signals with discontinuities
- Ideal solver semantics
- Choosing step sizes
- Discrete phase of execution
- Miscellaneous issues
  - Enabling vs. triggering guards
  - Order of reactions to simultaneous events
  - Nondeterministic state machines
  - Sampling discontinuous signals
  - Zeno behaviors

Issue 1: Enabling vs. Triggering Guards

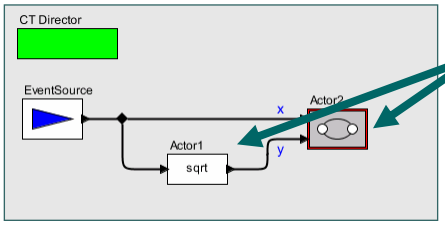
In Modal Models, guards on could have either of two semantic interpretations: *enabling* or *triggering*.



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.

HyVisual uses triggering semantics. Enabling semantics can be realized with an explicit Monte Carlo model.

Issue 2: Order of Reaction to Simultaneous Events



Semantics of a signal:

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

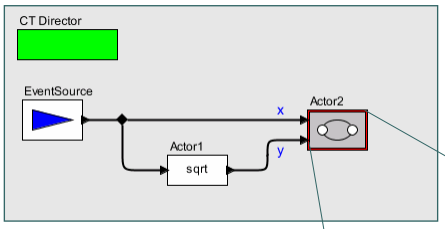
In HyVisual, every continuous-time signal has a value at $(t, 0)$ for any $t \in T$. This yields deterministic execution of the above model.

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

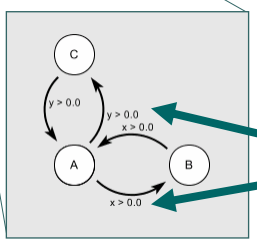
Simulink/Stateflow and HyVisual declare this to be deterministic, based on data precedences. Actor1 executes before Actor2.

Some formal hybrid systems languages declare this to be nondeterministic. We believe this is the wrong choice.

Issue 3: Nondeterministic State Machines



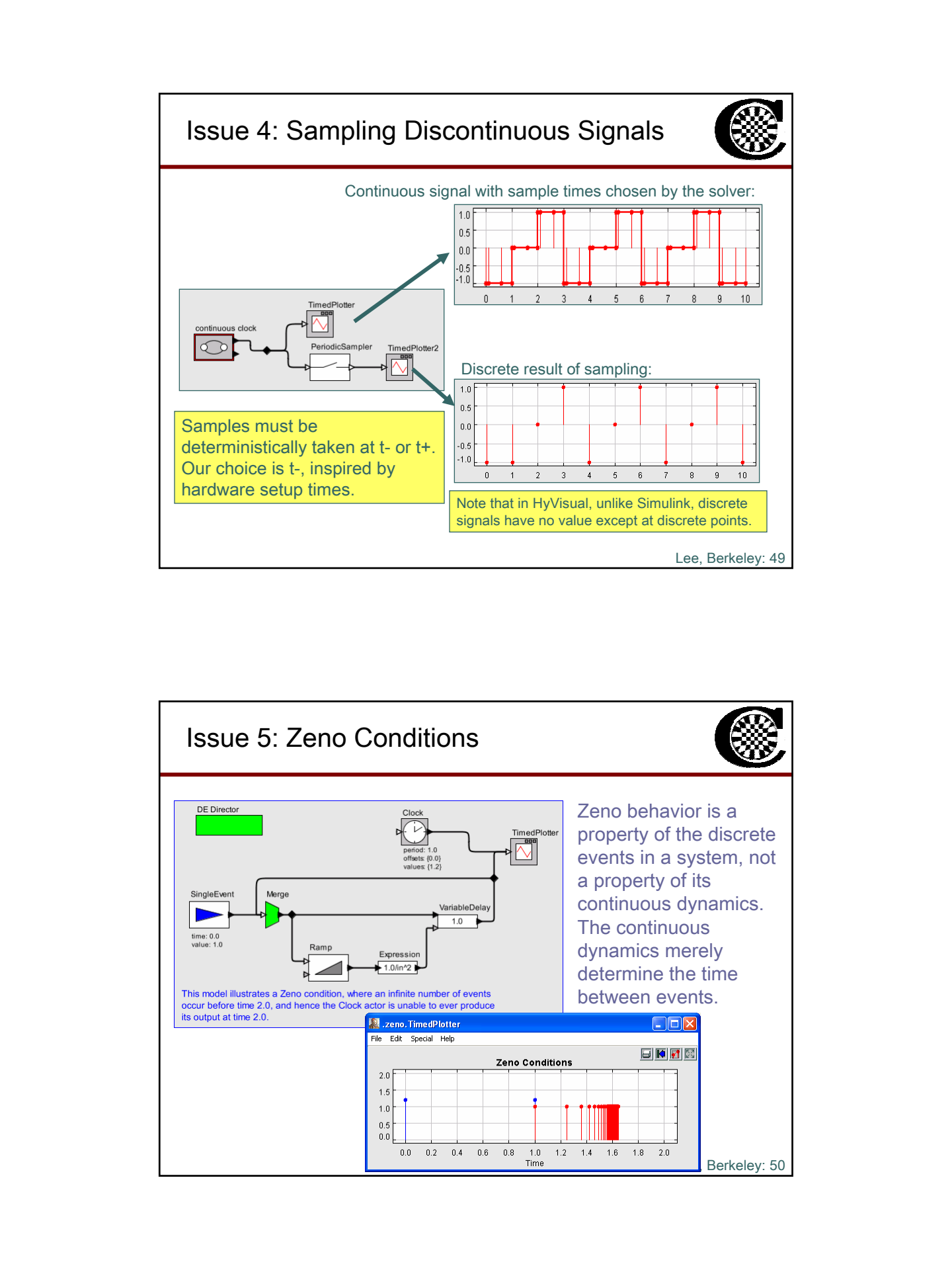
HyVisual supports explicit Monte Carlo models of nondeterminism.



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.

Issue 4: Sampling Discontinuous Signals

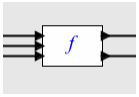


Issue 5: Zeno Conditions

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.

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.

A sufficient condition is that every feedback loop have a lower bound on its time delay. See [Lee 1999] for a review of this result, based on the Cantor Metric.

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. trapeziodal rule) are not delta causal, nor even strictly causal, so we have no assurance of a unique solution in feedback systems.

Summary

- Signals must be able to have multiple values at a time.
  - $x : T \times \mathbb{N} \rightarrow V$
- Actors must separate reactions to inputs from state updates
  - Supports event detection
  - Allows iterative step-size adjustment
- Compositionality
  - Need to be able to mix solvers
  - Need to be able to add hierarchy without changing behavior
- Many detail issues in designing executable hybrid systems:
  - Guards should trigger rather than enable transitions.
  - Precedence analysis is essential.
  - Nondeterminism is easily added with Monte Carlo methods
  - Sampling at discontinuitites needs to be well-defined.
  - Zeno conditions are a discrete event phenomenon

Open Source Software: HyVisual – Executable Hybrid System Modeling Built on Ptolemy II