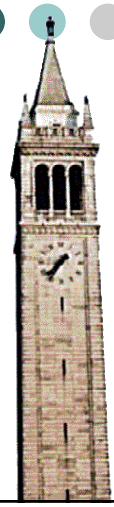


Hybrid System Modeling: Operational Semantics Issues

OMG Technical Meeting
Feb. 4, 2004
Anaheim, CA, USA

Edward A. Lee
Professor
UC Berkeley

Special thanks to
Jie Liu,
Xiaojun Liu,
Steve Neuendorffer, and
Haiyang Zheng.

Chess

Center for Hybrid and embedded software systems

Abstract

Chess, the Berkeley Center for Hybrid and Embedded Software Systems, has been studying the representation and execution of hybrid systems models. These models combine the discrete events of conventional software systems with the continuous dynamics of the physical world. Part of this effort has been an interaction with the DARPA MoBIES project (Model-Based Integration of Embedded Software), which has recently drafted a proposed "standard" for hybrid systems representation called HSIF, Hybrid System Interchange Format. In this presentation, I will be describe the issues that arise in the semantics of executable hybrid systems models. Fundamentally, computer systems are not capable of precise execution of hybrid system models because they cannot precisely realize the continuous dynamics. However, reasonable approximations are available, using for example numerical solvers for ordinary differential equations. However, these approximation techniques do not address the issues peculiar to hybrid systems, where discrete events can realize discontinuous behaviors in these ODEs. In this talk, I will outline the issues and how they have been addressed in Chess.

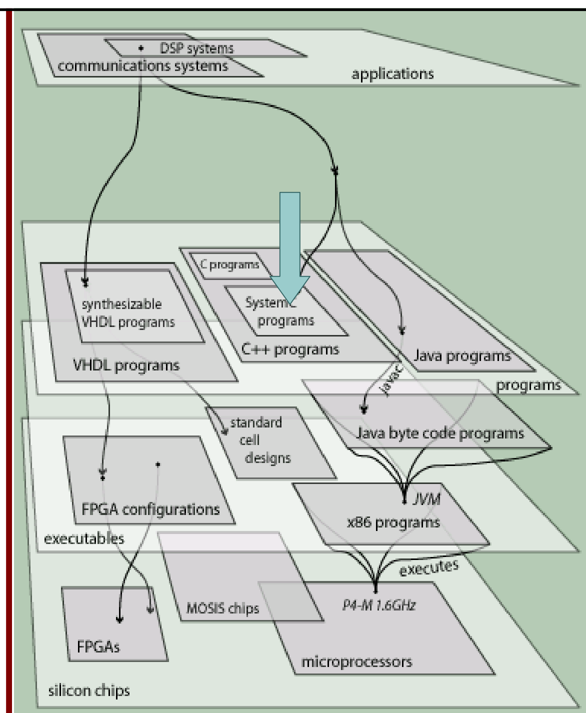
Focus on Hybrid & Embedded Software Systems

- Computational systems
  - but not first-and-foremost a computer
- Integrated with physical processes
  - sensors, actuators
- Reactive
  - at the speed of the environment
- Heterogeneous
  - hardware/software, mixed architectures
- Networked
  - adaptive software, shared data, resource discovery

Model-Based Design

Recall from the Previous talk:

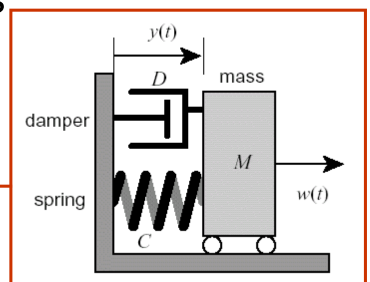
Model-based design is specification of designs in platforms with “useful modeling properties.”

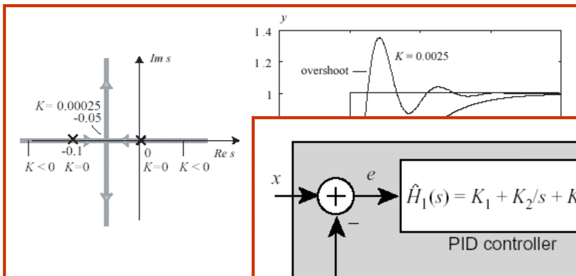


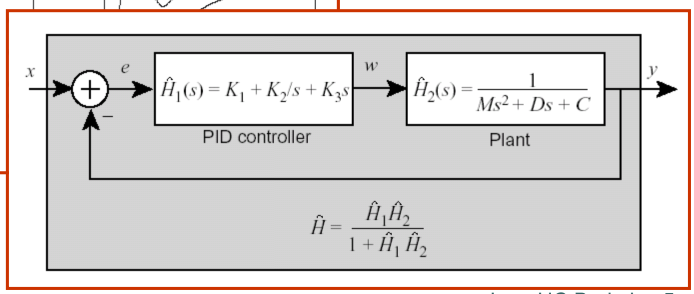
"Useful Modeling Properties" for Embedded Systems

Example: Control systems:

- Continuous dynamics
- Stability analysis

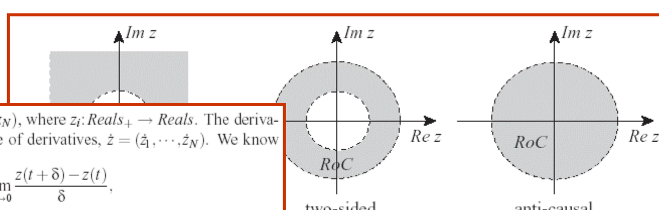






Discretized Model A Small Step Towards Software

- Numerical integration techniques provided sophisticated ways to get from the continuous idealizations to computable algorithms.
- Discrete-time signal processing techniques offer the same sophisticated stability analysis as continuous-time methods.
- But it's not accurate for software controllers



In general, z is an N-tuple, $z = (z_1, \cdots, z_N)$, where $z_i\colon Reals_+ \to Reals$. The derivative of an N-tuple is simply the N-tuple of derivatives, $\dot{z} = (\dot{z}_1, \cdots, \dot{z}_N)$. We know from calculus that

$$\dot{z}(t) = \frac{dz}{dt} = \lim_{\delta \to 0} \frac{z(t+\delta) - z(t)}{\delta},$$

and so, if $\delta > 0$ is a small number, we can approximate this derivative by

$$\dot{z}(t) \approx \frac{z(t+\delta) - z(t)}{\delta}.$$

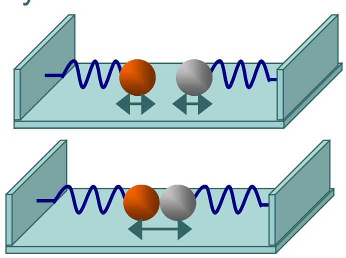
Using this for the derivative in the left-hand side of (5.50) we get

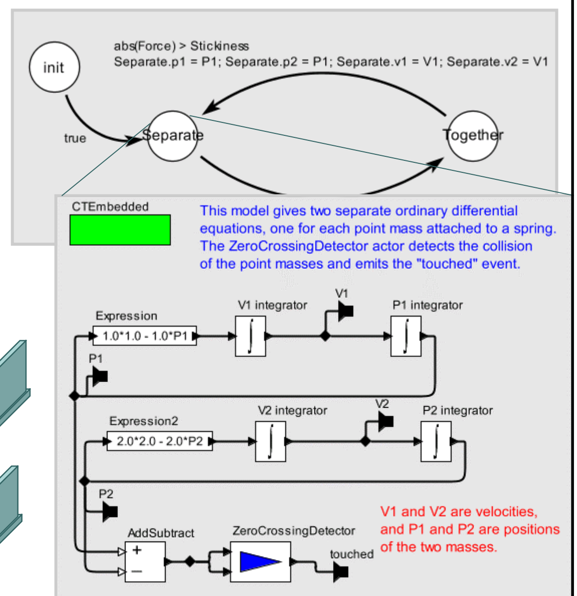
$$z(t+\delta) - z(t) = \delta g(z(t), v(t)). \tag{5.51}$$

Hybrid Systems – A Bigger Step Towards Software

Combine:

- finite-state automata
- classical models of continuous or discrete-time dynamics

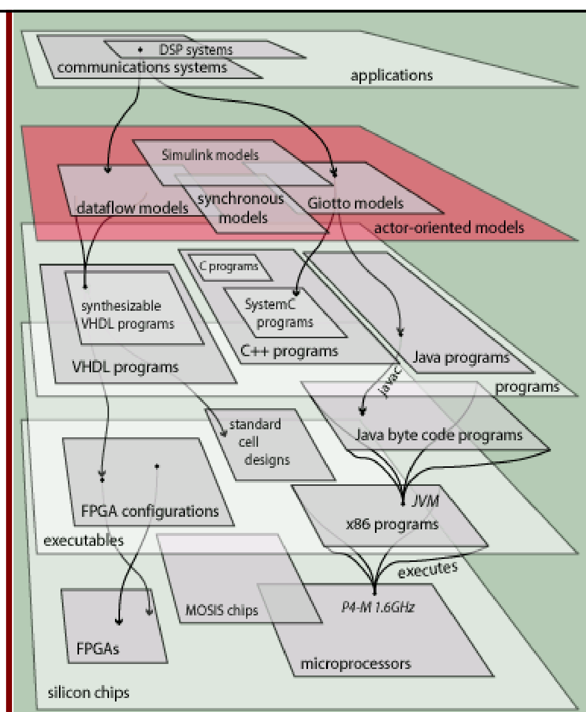




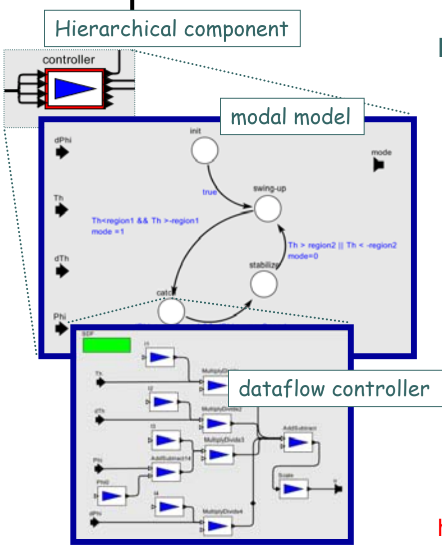
Actor-Oriented Platforms

Recall from the Previous talk:

Actor oriented models compose concurrent components according to a model of computation.



Ptolemy II – Our Laboratory



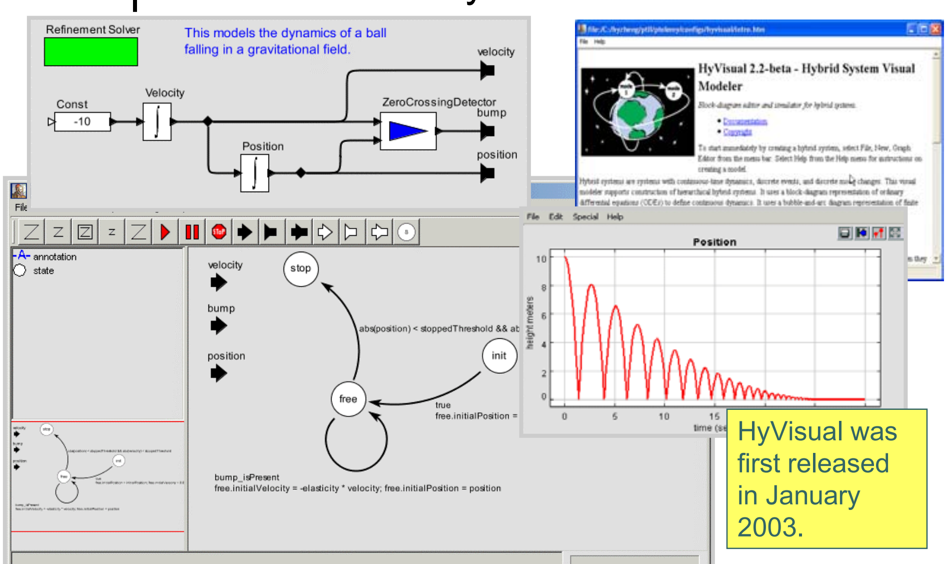
example Ptolemy II model: hybrid control system

Ptolemy II:

Our current framework for experimentation with actor-oriented design, concurrent semantics, visual syntaxes, and hierarchical, heterogeneous design.

http://ptolemy.eecs.berkeley.edu

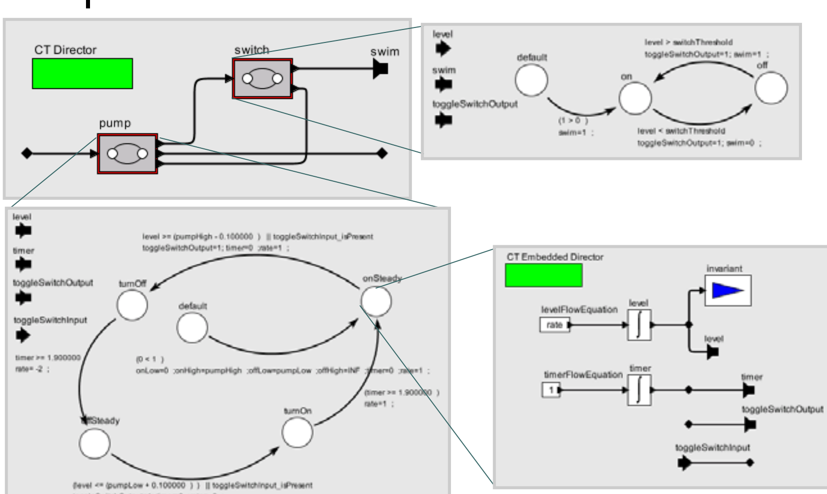
HyVisual – Hybrid System Modeling Tool Based on Ptolemy II



Operational Semantics of Hybrid Systems (How to Build Simulators)

- If you are going to rely on simulation results, then you need an operational semantics.
  - Hybrid system semantics tend to be denotational.
- A simulator cannot ignore nondeterminism.
  - It is incorrect to choose one trajectory.
  - Creating deterministic models must be easy.
  - Nondeterministic models must be explored either exhaustively or using Monte Carlo methods.
  - *Must* avoid unnecessary nondeterminism.
- Should not use continuous-time models to represent discrete behaviors.
  - Inaccurate for software.
  - Heterogeneous models are better.

View Hybrid Systems as Networks of Automata



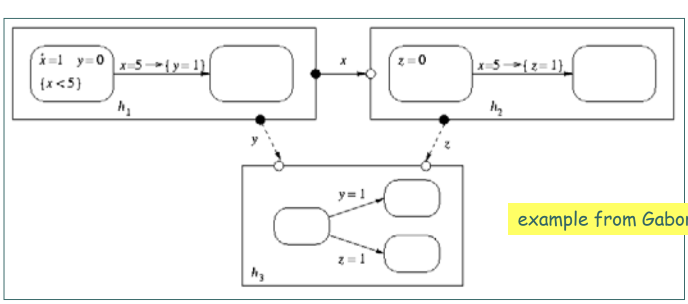
The key question becomes: What is the semantics for the interaction between automata?

Many Interaction Semantics Between Automata Have Been Tried

- Asynchronous
  - Promela (specification language for Spin)
  - SDL
  - Ptolemy II (PN+FSM, DE+FSM)
- Synchronous w/ fixed point
  - Esterel
  - Simulink
  - Ptolemy II (SR+FSM)
- Synchronous w/out fixed point
  - Statecharts
  - Giotto
  - Ptolemy II (SDF+FSM)
- Continuous time
  - Simulink + Stateflow
  - Ptolemy II (CT+FSM)
- Discrete time
  - Teja

Context of the Discussion

- DARPA/MoBIES Effort to Standardize: Hybrid System Interchange Format: HSIF
- HSIF allows modeling of Networks of Hybrid Automata
- Automata interact via ***signals*** (synchronous semantics) and ***global variables*** (unrestricted)



example from Gabor Karsai, Vanderbilt

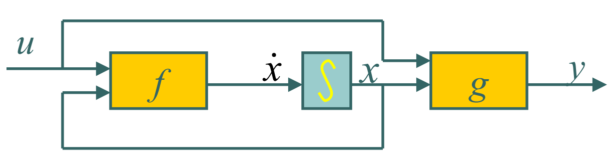
Some Semantics Questions

- What automata can be expressed?
  - nondeterministic, guard expression language, actions, …
- How are transitions in distinct automata coordinated?
  - synchronous, time-driven, event-driven, dataflow, …
  - can outputs and updates be separated?
- What can automata communicate?
  - messages, events, triggers
- How is communication carried out?
  - synchronous, rendezvous, buffered, lossy, …
- How are continuous variables shared?
  - global name space, scoping, mutual exclusion, …
- What is the meaning of directed cycles?
  - fixed point, error, infinite loop, …
- What is the meaning of simultaneous events?
  - secondary orderings, such as data precedences, priorities, …

Interaction Between ODE Solvers and State Machine Dynamics

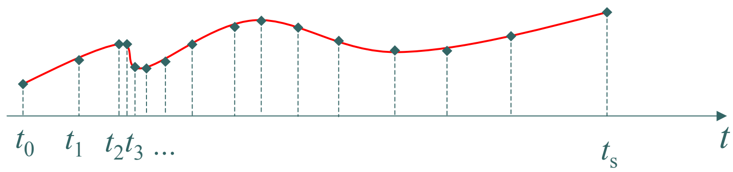
Modeling continuous dynamics using
Initial Value Ordinary Differential Equations:

$$\boxed{\begin{aligned} \dot{x} &= f(x,u,t) \\ x(t_0) &= x_0 \\ y &= g(x,u,t) \end{aligned}}$$

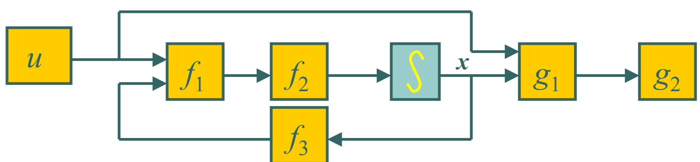


ODE Solvers

- Numerical solution of the ODE on discrete time points.
- Implementing ODE solvers by token passing
- Evaluate *f* and *g* by firing a sorted sequence of components.

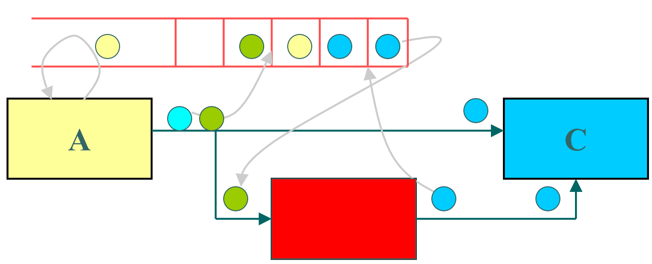


Step sizes are dynamically determined!



Executing Discrete Event Systems

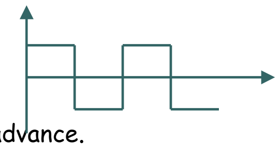
- Global notion of time
- event = (time_tag, data_token)
- Event-driven execution
- Global event queue, sorting events in their chronological order
- Components are causal
- Components can schedule "refires" by producing pure events.



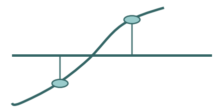
Mixing The Two Means Dealing with Events In Continuous-Time Signals

Breakpoint Handling:

- Predictable Breakpoints:
  - known beforehand.
  - Register to a Breakpoint Table in advance.
  - Use breakpoints to adjust step sizes.

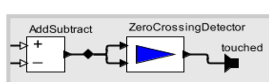


- Unpredictable Breakpoints:
  - Prediction is not accurate enough.
  - Check after each integration step.
  - Refine the last step size if a breakpoint is missed.

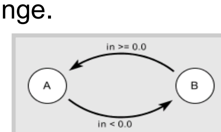


Transitions of an FSM Are Discrete Events

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

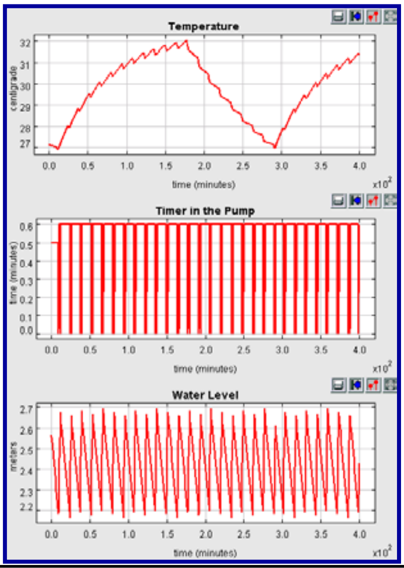


- Semantics of transitions: can either *enable* a mode change or trigger a mode change.

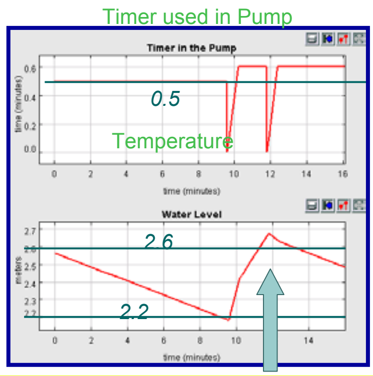


- Under enabling: deterministic model becomes nondeterministic if simulator takes steps that are too large.
- Also under enabling: invariants may be violated due to failure to take mode transitions on time.

Guards Enabling Transitions is the Wrong Answer!

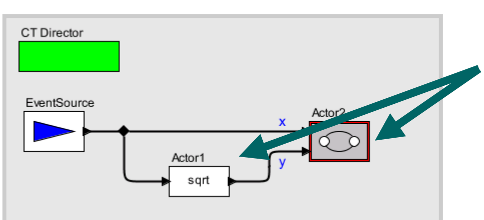


Can yield values that are conceptually impossible in the model, purely as an artifact of the chosen step size.



In this example, overshoot violates invariants

Simultaneous Events: The Order of Execution Question



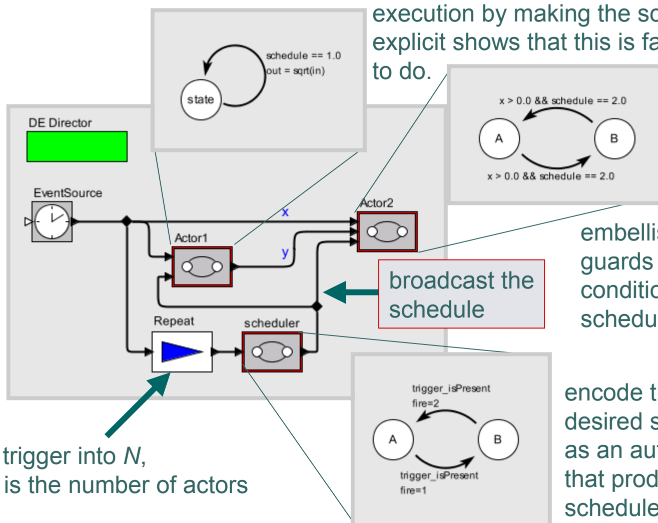
Given an event from the event source, which of these should react first? Nondeterministic? Data precedences?

Simulink/Stateflow and the Ptolemy II CT domain declare this to be deterministic, based on data precedences. Actor1 executes before Actor2.

Many formal hybrid systems languages (with a focus on verification) declare this to be nondeterministic.

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.

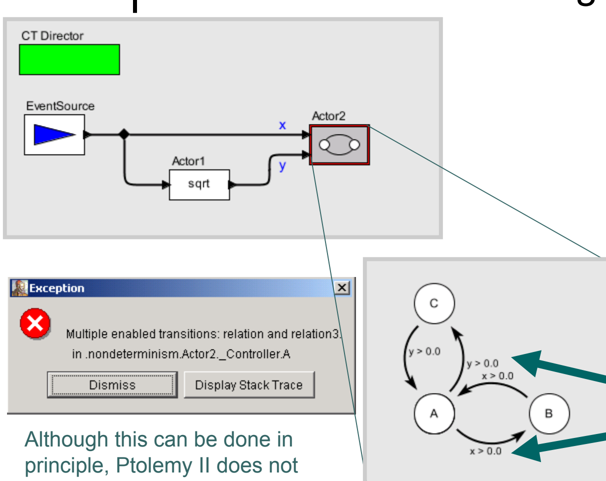


embellish the guards with conditions on the schedule

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



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

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

Nondeterministic Ordering

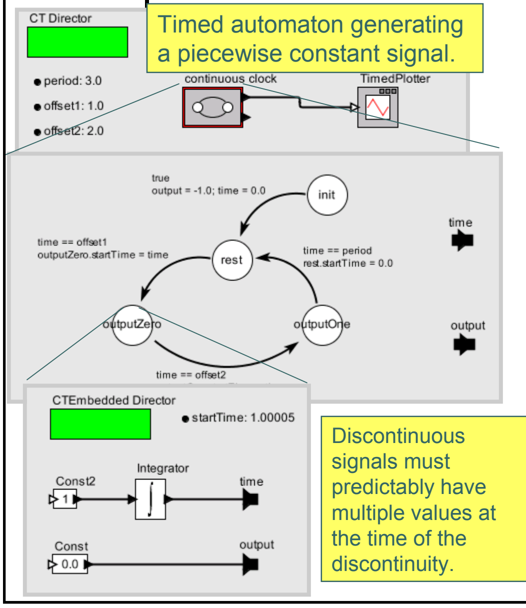
- In favor
  - Physical systems have no true simultaneity
  - Simultaneity in a model is artifact
  - Nondeterminism reflects this physical reality
- Against
  - It surprises the designer
    - counters intuition about causality
  - It is hard to get determinism
    - determinism is often desired (to 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!

More Semantics Questions: How to Get Predictable Execution

- Discontinuous signals must have zero transition times.
  - Precise transition times.
  - Accurate model of Zeno conditions.
  - Avoid unnecessary nondeterminism.
- Discrete signals should have values only at discrete times
  - Accurately heterogeneous model (vs. continuous approximation)
- Sampling of discontinuous signals must be well-defined.
  - Avoid unnecessary nondeterminism.
- Transient states must be active for zero time.
  - Properly represent glitches.

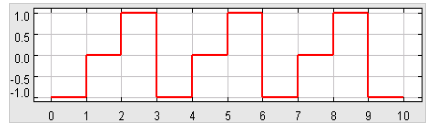
Discontinuous Signals

Timed automaton generating a piecewise constant signal.

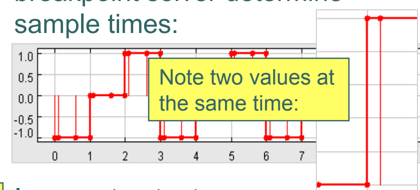


Discontinuous signals must predictably have multiple values at the time of the discontinuity.

Correct output:

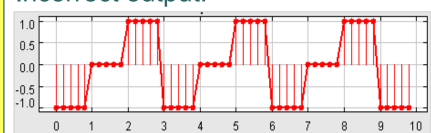


RK 2-3 variable-step solver and breakpoint solver determine sample times:



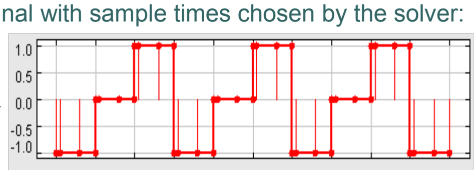
Note two values at the same time:

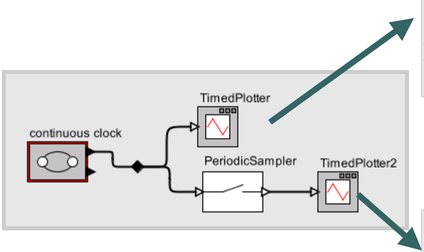
Incorrect output:



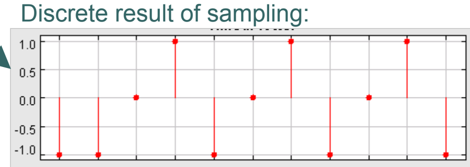
Sampling Discontinuous Signals

Continuous signal with sample times chosen by the solver:





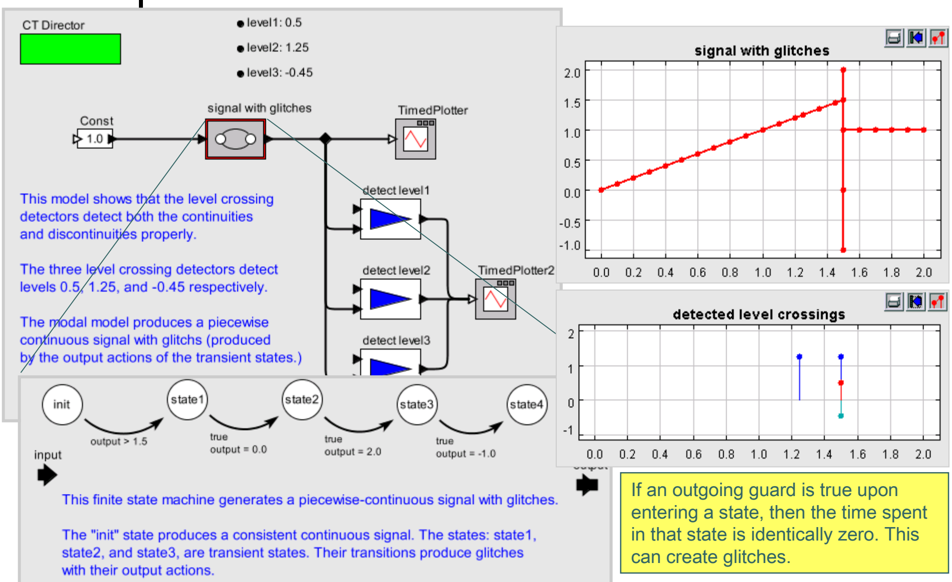
Discrete result of sampling:



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

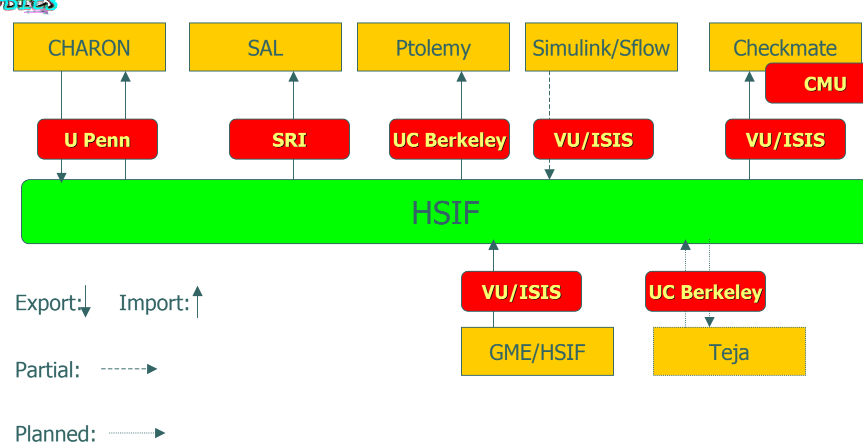
Note that in Ptolemy II, unlike Simulink, discrete signals have no value except at discrete points.

Transient States and Glitches



If an outgoing guard is true upon entering a state, then the time spent in that state is identically zero. This can create glitches.

Status of HSIF: Limited Tool Interchange



courtesy of Gabor Karsai, Vanderbilt

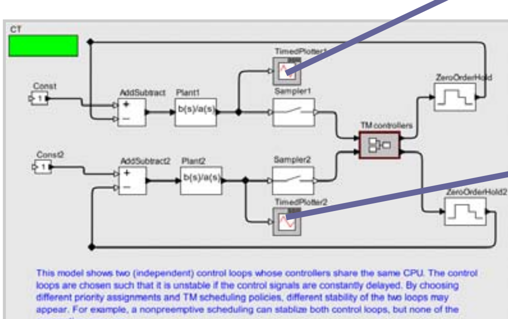
Personal Experience with HSIF

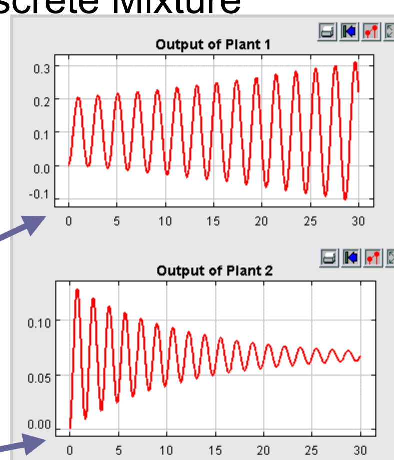
- Models exchanged between the tools had limited value:
  - Imported models had enough translation applied that little intuition remained about the model.
  - Exporting models is only practical if the exporting framework exactly matches the HSIF semantics.
- Hybrid systems don't solve the whole problem anyway.
- More work is needed…

Caveat: Hybrid Systems are Not the Only Useful Continuous/Discrete Mixture

An example, due to Jie Liu, has two controllers sharing a CPU under an RTOS. Under preemptive multitasking, only one can be made stable (depending on the relative priorities). Under non-preemptive multitasking, both can be made stable.

Hybrid systems theory does not deal well with this.



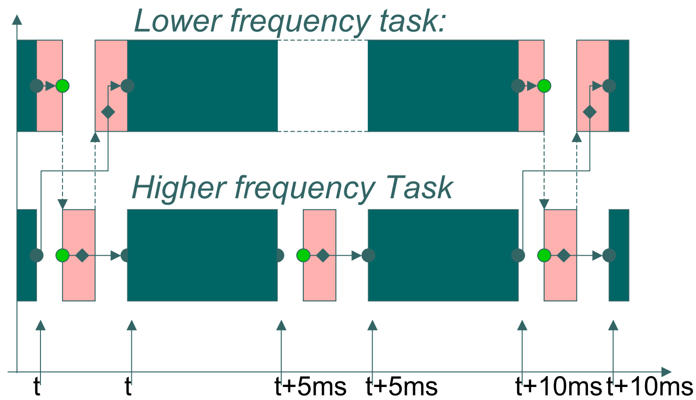


Modeling multitasking with hybrid systems is extremely awkward.

Alternatives Give Clean Temporal Semantics to Software: e.g. Giotto

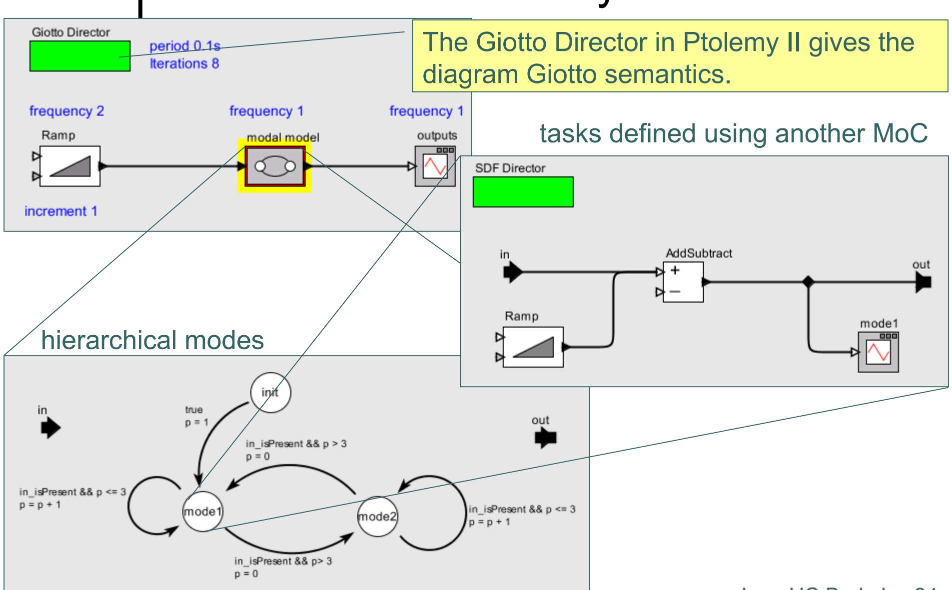
Giotto – Periodic Hard-Real-Time Tasks with Precise Mode Changes.

Deterministic task interaction.

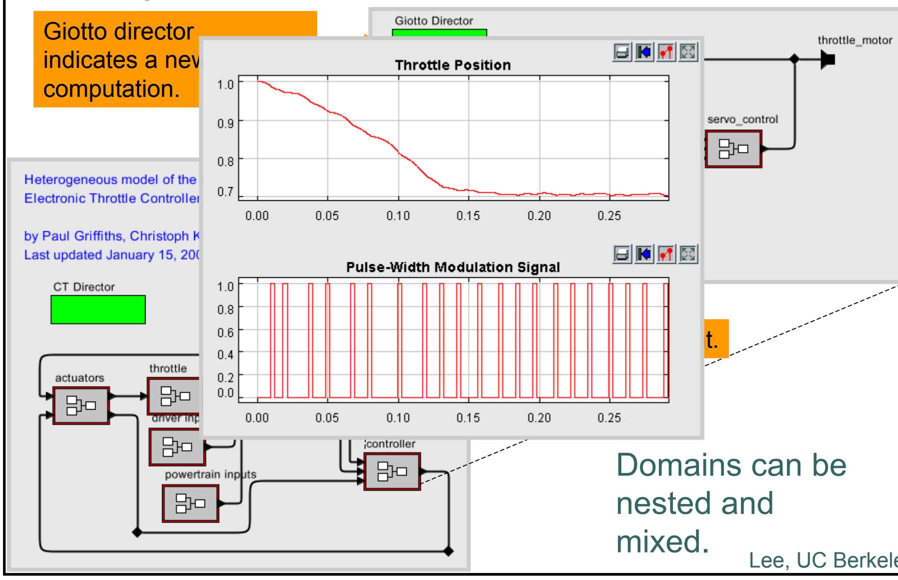


- Giotto compiler targets the E Machine/S Machine
- Created by Tom Henzinger and colleagues
- Giotto model of computation also implemented in Ptolemy II

Giotto with a Visual Syntax

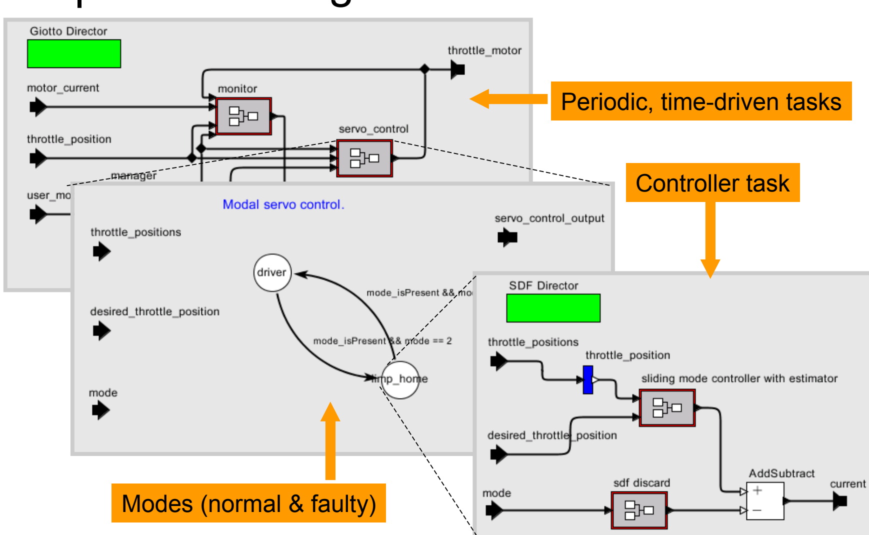


Design Pattern: Periodic/Time-Driven Inside Continuous Time

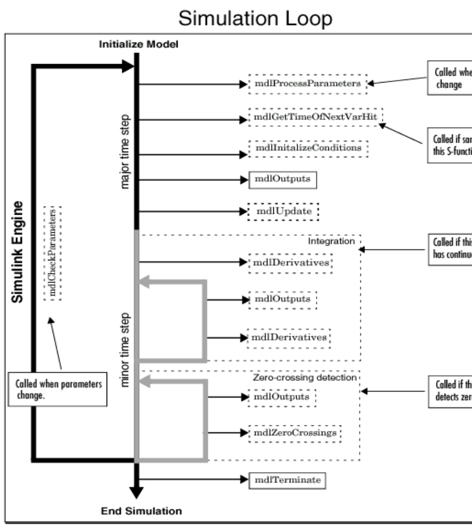


Domains can be nested and mixed.

Nesting Giotto With State Machine for Modeling Faults



Simulink With Real-Time Workshop Has Similar Semantics



- continuous time
- discrete actors are logically instantaneous
- separation of output/update methods to support algebraic loops, integration, and zero-crossing detection
- output method invoked many times
- multitasking mode for periodic discrete-time tasks.
- multitasking mode requires Giotto-like delayed output commit

image from *Writing S-Functions*, version 4, The MathWorks

Conclusion

- Modeling hybrid systems correctly is subtle
- There are other formalisms for discrete/continuous mixtures
- Standardization will be challenging
- see http://ptolemy.eecs.berkeley.edu