Modal Models in Ptolemy

Edward A. Lee
Stavros Tripakis
UC Berkeley

Workshop on Equation-Based Object-Oriented Modeling Languages and Tools

3rd International Workshop on
Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT)

Oslo, Norway
Oct 3, 2010, in conjunction with MODELS

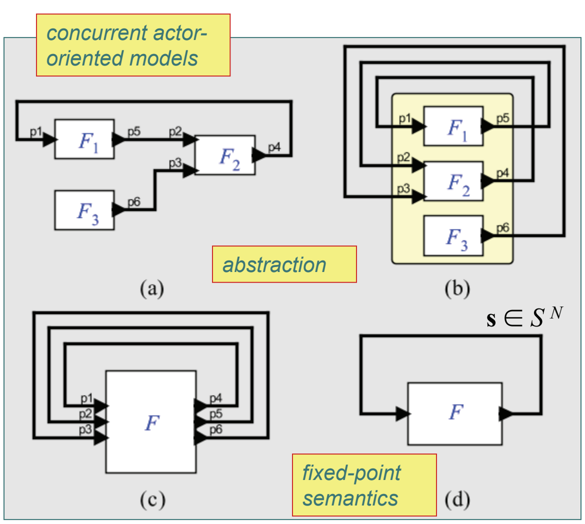
Online

Online interactive versions of most of the examples in this talk can be accessed by clicking on the figures in the paper listed here:
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-151.html

Influences for This Work

- Statecharts [Harel 87]
- Argos [Maraninchi 91]
- Esterel [Berry & Gonthier 92]
- Abstract state machines [Gurevich 93]
- Hybrid systems [Puri & Varaiya 94, Henzinger 99]
- Timed automata [Alur & Dill 94]
- SyncCharts [Andre 96]
- I/O Automata [Lynch 96]
- *Charts [Girault, Lee, & Lee 99]
- UML State machines

What does this have to do with Equational Models?

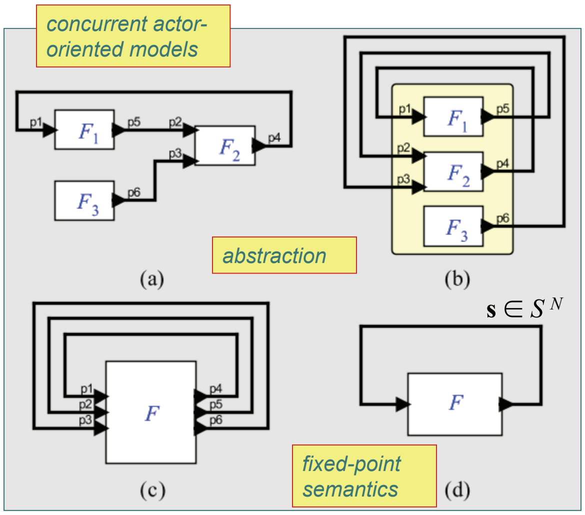


Fixed-point semantics for a rich variety of concurrent models of computation:

- Fixed-point semantics: $\mathbf{s} = F(\mathbf{s})$
- Dataflow models: $\mathbf{s}$ is a tuple of sequences.
- Synchronous/reactive models: $\mathbf{s}$ is a tuple of values, and $F = F_i$ varies in each tick i.
- Discrete-event models: SR model where each tick occurs at a time τ_i in a model of time.
- Continuous-time models: DE models where a solver chooses the values of τ_i.

[Lee & Zheng, EMSOFT 07]

What does this have to do with Equational Models?

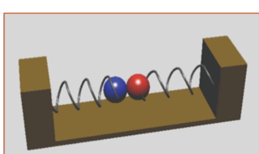


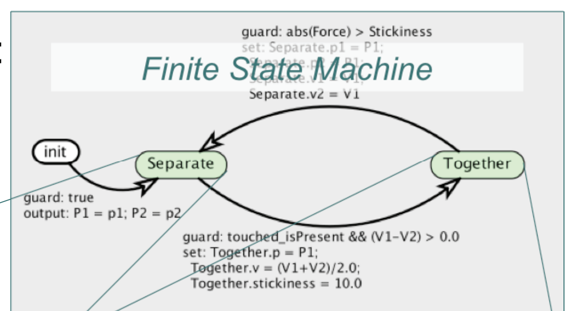
For SR, DE, and CT models, the function *F* changes over time.

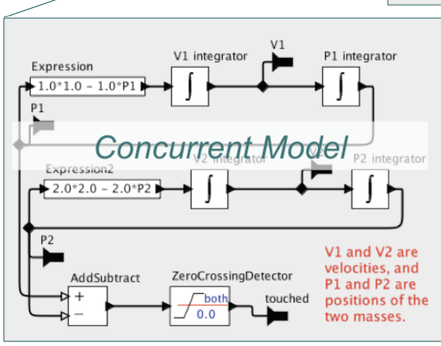
This talk gives a semantics to these changes using the notion of *modal models*.

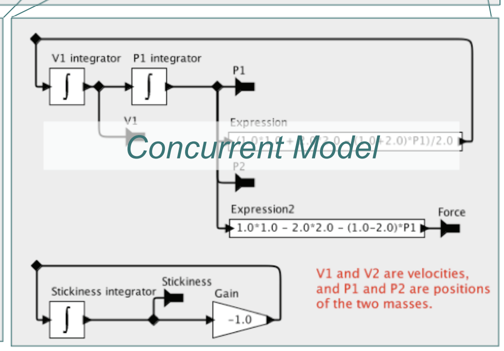
A major goal is a clean semantics for hybrid systems.

Motivating Example: Hybrid System

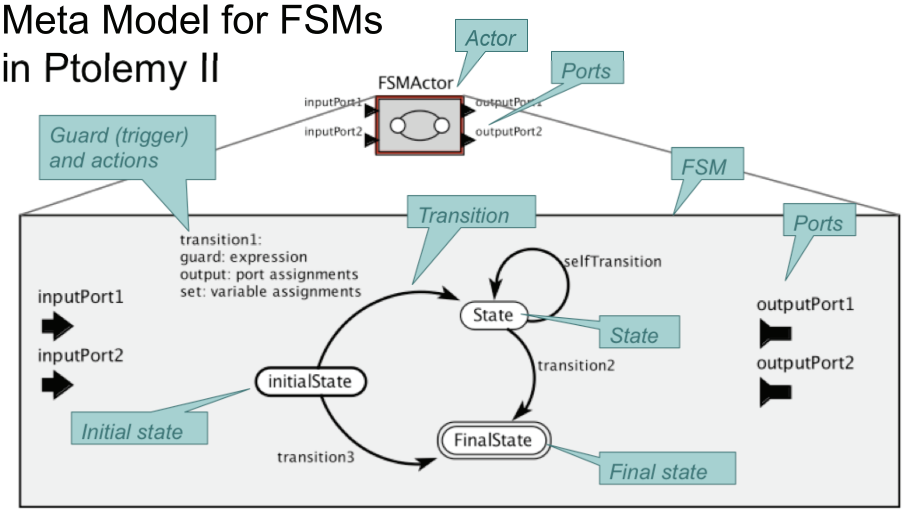








Meta Model for FSMs in Ptolemy II

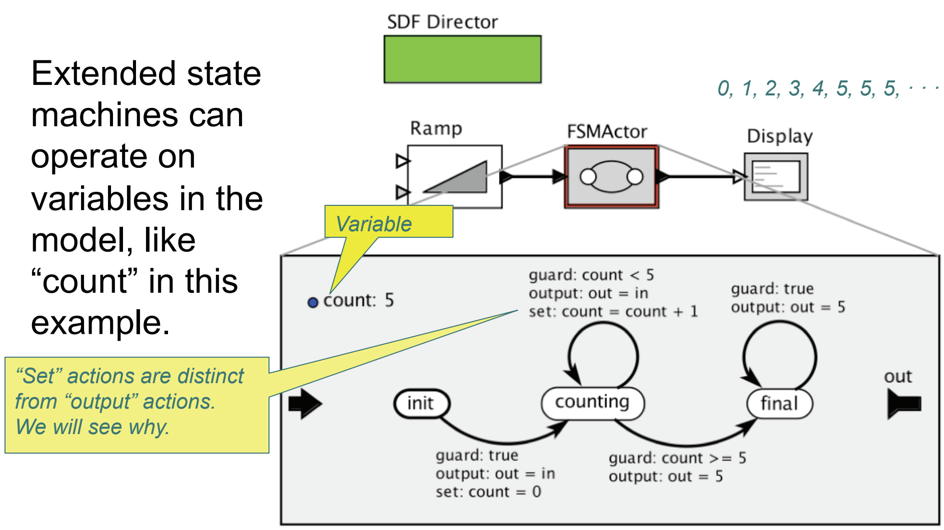


- Initial state indicated in bold
- Guards are expressions that can reference inputs and variables
- Output values can be functions of inputs and variables
- Transition can update variable values ("set" actions)
- Final state terminates execution of the actor

Extended State Machines

Reference and manipulate variables on guards and transitions.

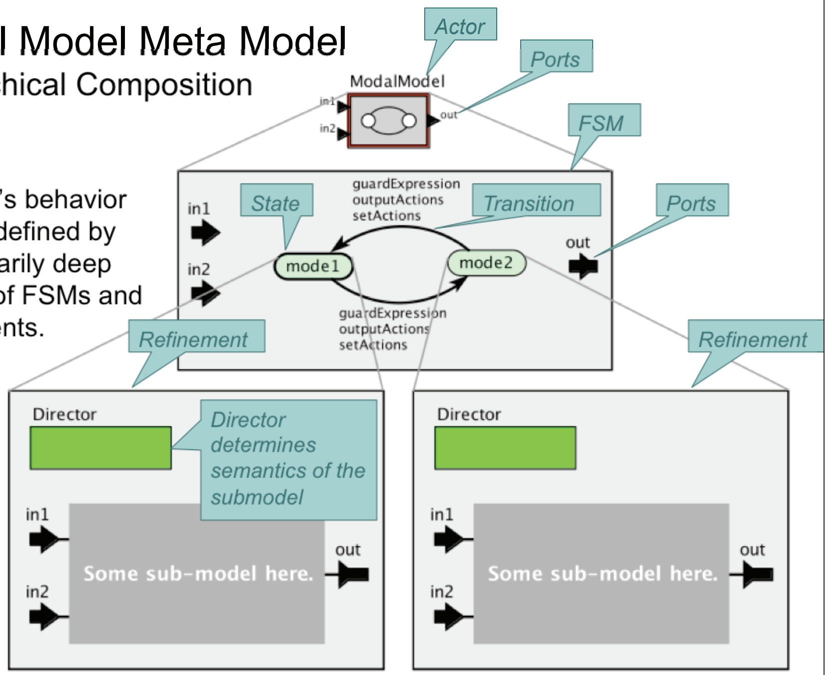
Extended state machines can operate on variables in the model, like "count" in this example.



Modal Model Meta Model

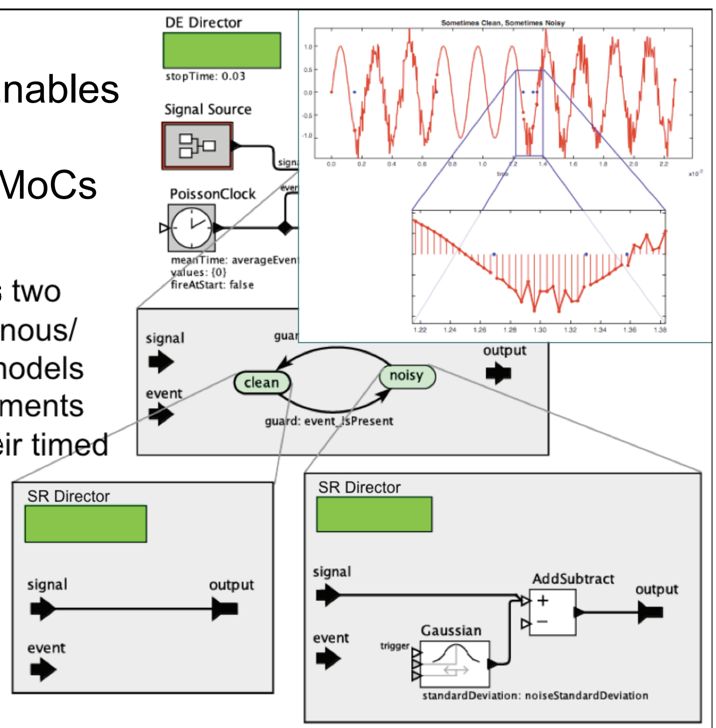
Hierarchical Composition

An actor's behavior may be defined by an arbitrarily deep nesting of FSMs and refinements.



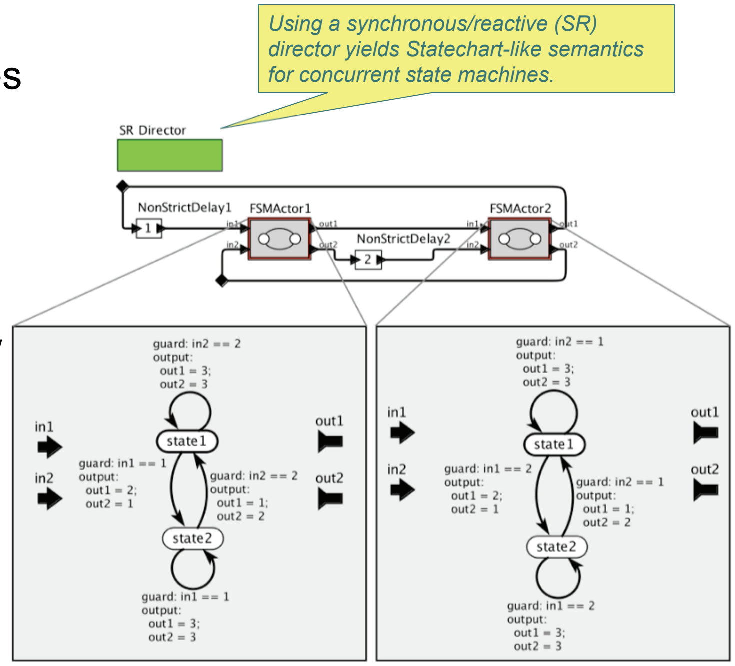
Ptolemy II Enables Hierarchical Mixtures of MoCs

This model has two simple synchronous/ reactive (SR) models as mode refinements and models their timed environment using a discrete-event (DE) director



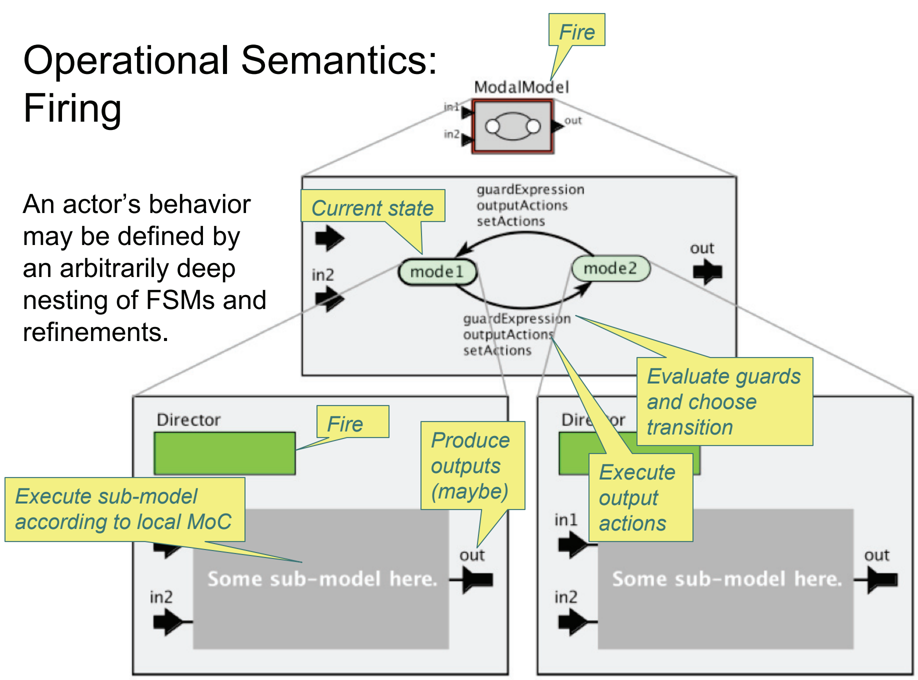
AND states

Here, two FSMs are composed under a synchronous/reactive director, resulting in Statecharts-like AND states.



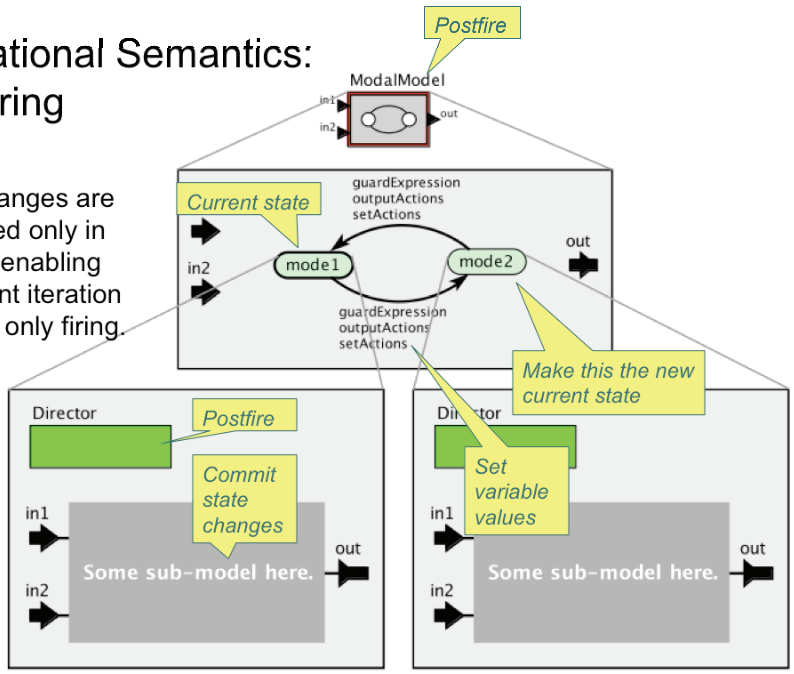
Operational Semantics: Firing

An actor's behavior may be defined by an arbitrarily deep nesting of FSMs and refinements.



Operational Semantics: Postfiring

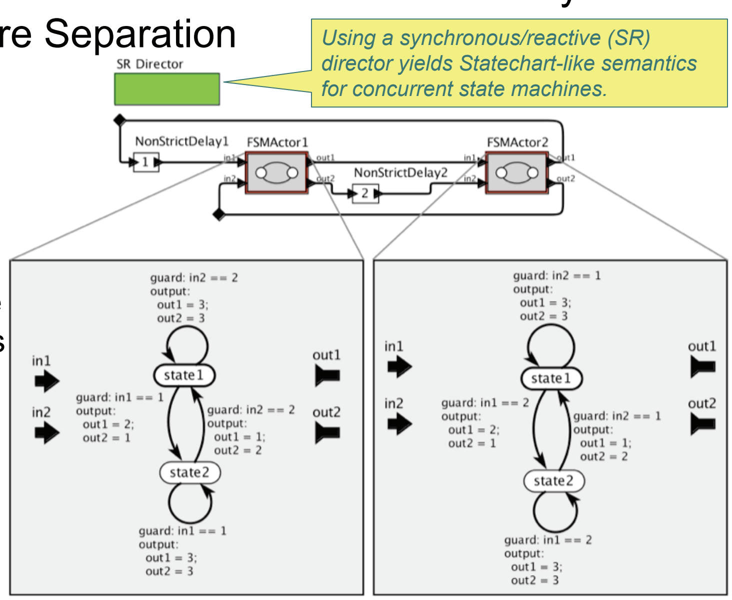
State changes are committed only in postfire, enabling fixed point iteration by using only firing.



Fixed-Point Semantics of SR is Enabled by Fire/Postfire Separation

Result is a constructive semantics.

The example here requires multiple firings of the FSM actors to converge to the least fixed point.



Directors Benefiting from Fire/Postfire Separation (which we call the Actor Abstract Semantics)

- Synchronous/Reactive (SR)
  - Execution at each tick is defined by a least fixed point of monotonic functions on a finite lattice, where bottom represents "unknown" (getting a constructive semantics)
- Discrete Event (DE)
  - Extends SR by defining a "time between ticks" and providing a mechanism for actors to control this. Time between ticks can be zero ("superdense time").
- Continuous
  - Extends DE with a "solver" that chooses time between ticks to accurately estimate ODE solutions, and fires all actors on every tick.

[Lee & Zheng, EMSOFT 07]

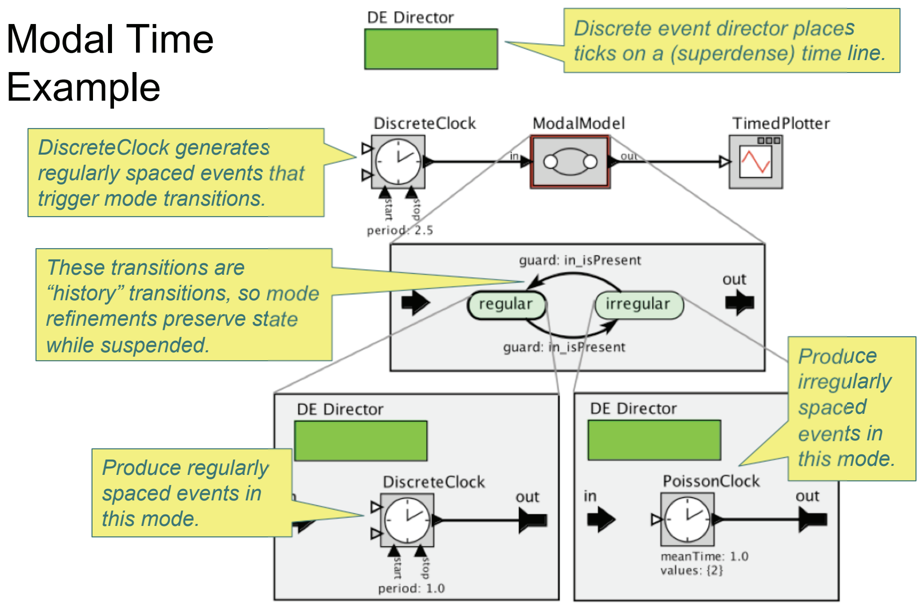
The Modal Model Muddle

It's about time

After trying several variants on the semantics of modal time, we settled on this:

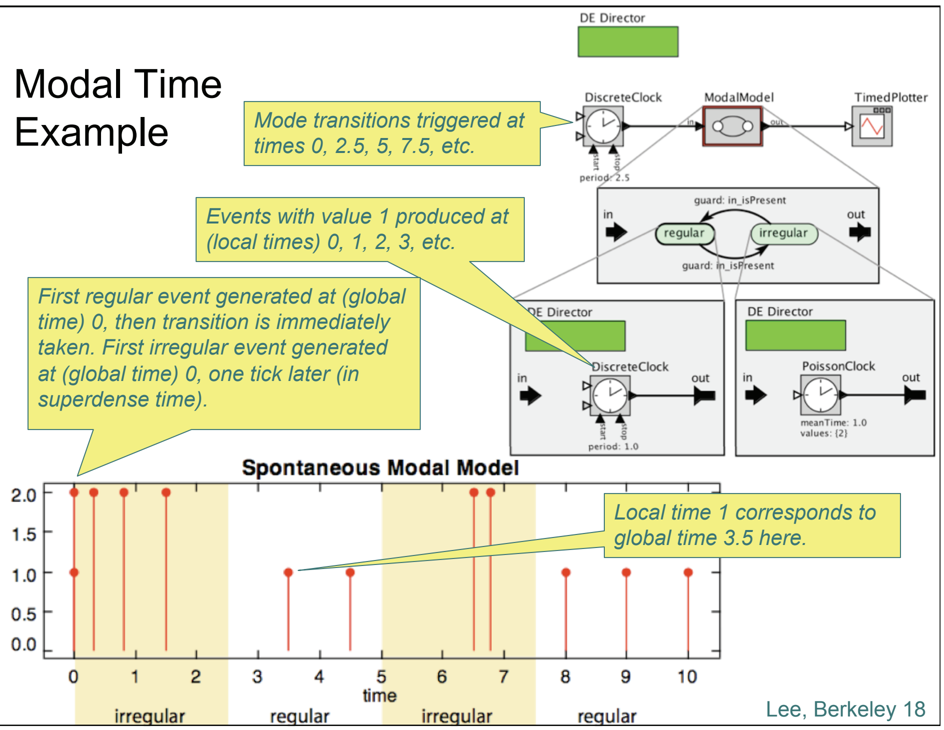
A mode refinement has a *local* notion of time. When the mode refinement is inactive, local time does not advance. Local time has a monotonically increasing gap relative to global time.

Modal Time Example



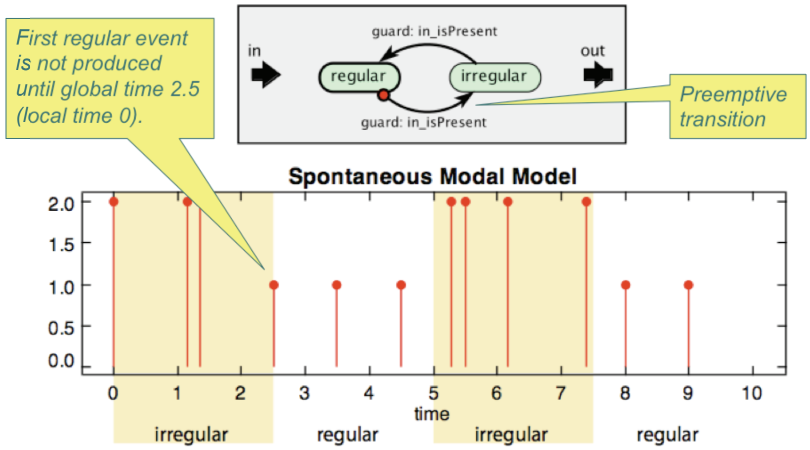
Lee, Berkeley 17

Modal Time Example

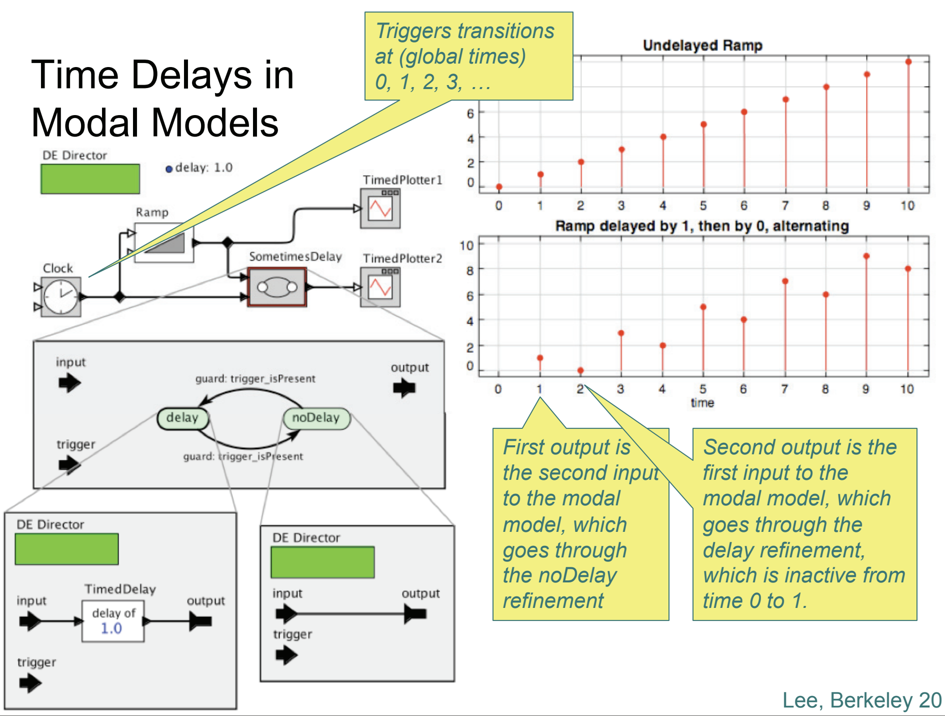


Lee, Berkeley 18

Variant using Preemptive Transition



Time Delays in Modal Models



Variants for the Semantics of Modal Time that we Tried or Considered, but that Failed

- Mode refinement executes while "inactive" but inputs are not provided and outputs are not observed.
- Time advances while mode is inactive, and mode refinement is responsible for "catching up."
- Mode refinement is "notified" when it has requested time increments that are not met because it is inactive.
- When a mode refinement is re-activated, it resumes from its first missed event.

All of these led to some very strange models…

Final solution: Local time does not advance while a mode is inactive. Growing gap between local time and global time.

Formalization (Detailed in the Paper)

- Actor:

States Initial state Inputs Outputs

$$A = (S, s_0, I, O, F, P)$$

- Output function ("fire"):

$$F : S \times I \times \mathrm{N} \to O$$

- State-update function ("postfire"):

Resolves potential non-determinism

$$P : S \times I \times \mathrm{N} \to S$$

Semantics – untimed

- Set of traces:

$$s_0 \xrightarrow{x_0, y_0} s_1 \xrightarrow{x_1, y_1} s_2 \xrightarrow{x_2, y_2} \cdots$$

- such that for all i, there exists j, s.t.:

$$y_i = F(s_i, x_i, j)$$
$$s_{i+1} = P(s_i, x_i, j)$$

Semantics – timed

- States include special *timer* variables:
  - Can be suspended (“frozen”, inactive) and resumed (active)
  - Expire when they reach 0
- Set of timed traces:

Delays

$$s_0 \xrightarrow{x_0, y_0, d_0} s_1 \xrightarrow{x_1, y_1, d_1} s_2 \xrightarrow{x_2, y_2, d_2} \cdots$$

- such that for all i, there exists j, s.t.:

Decrement active timers by d_i

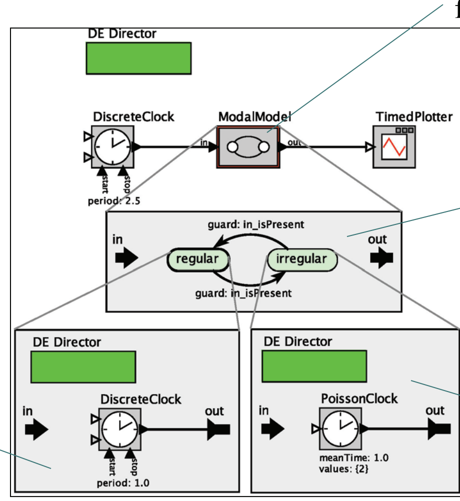
$$y_i = F(s_i, x_i, j)$$
$$s_{i+1} = P(s_i - d_i, x_i, j)$$
$$d_i \leq \min\{v \mid v \text{ is the value of an active timer in } s_i\}$$

Composition and heterogeneity

- Modular semantics:
  - Given a composite actor A, with sub-actors A1, A2, …
  - the F and P functions for A are defined from the Fi, Pi, functions of sub-actors Ai.
- How F and P are defined depends on the director used in A
  - Directors = composition operators
- Heterogeneity:
  - Different directors implement different composition models

Giving semantics to modal models

Goal: define F, P functions for the modal model

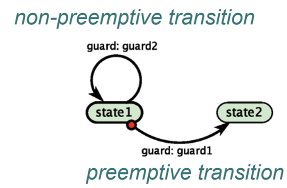


F_c, P_c functions already defined for the "controller" automaton

F_1, P_1 functions already defined for this refinement

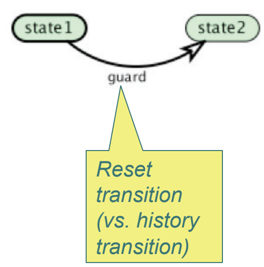
F_2, P_2 functions already defined for this refinement

Rough description of semantics

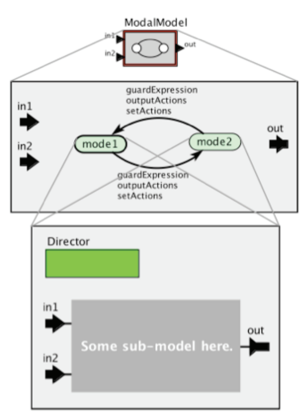


- Given current controller state s_i :
- If no outgoing transitions from s_i are enabled:
  - Use F_i and P_i to compute F and P
- If preemptive outgoing transitions from s_i are enabled:
  - Use the actions of these transitions to compute F and P
- If only non-preemptive outgoing transitions from s_i are enabled:
  - First fire refinement, then transition, i.e.:
    - F is the composition of F_i and the output action of a transition
    - P is the composition of P_i and the state update action of a transition
- Timers of refinements suspended and resumed when exiting/entering states
- Details in the paper

More Variants of Modal Models Supported in Ptolemy II

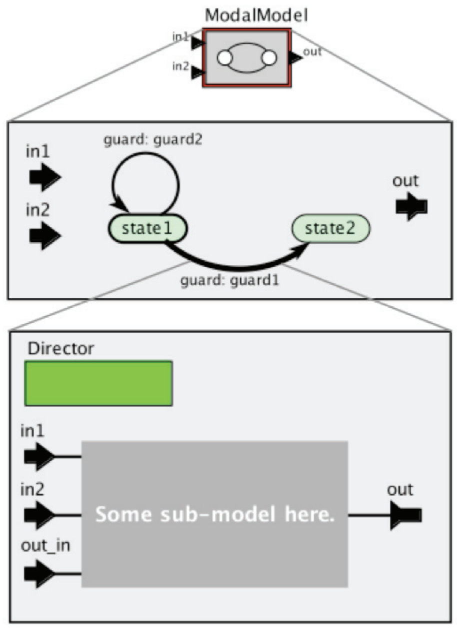


- Transition may be a reset transition
  - Destination refinement is initialized
- Multiple states can share a refinement
  - Facilitates sharing internal actor state across modes
- A state may have multiple refinements
  - Executed in sequence (providing imperative semantics)



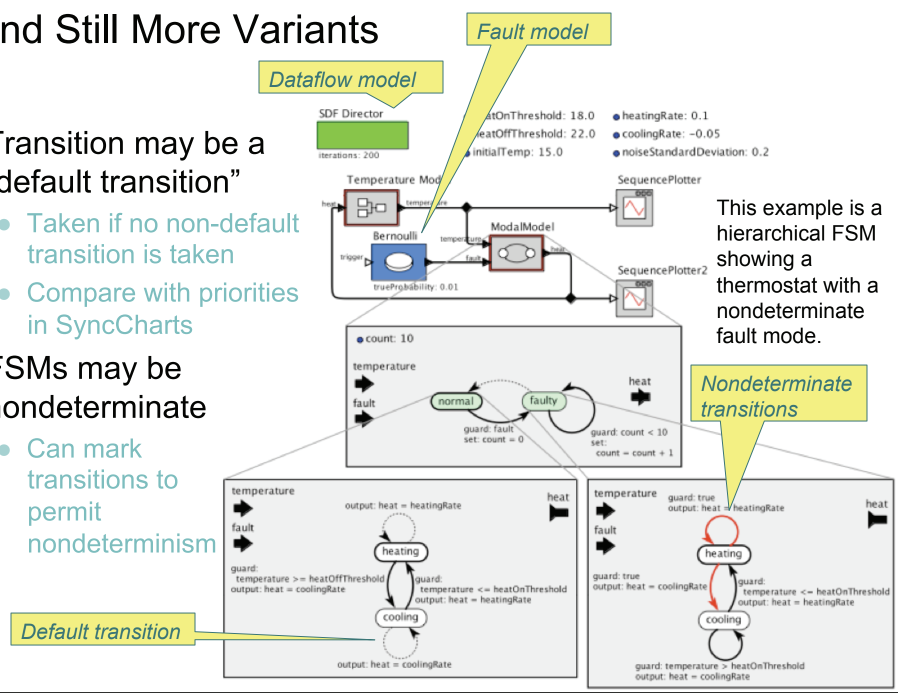
Still More Variants

- Transition may have a refinement
  - Refinement is fired when transition is chosen
  - Postfired when transition is committed
  - Time is that of the environment



And Still More Variants

- Transition may be a "default transition"
  - Taken if no non-default transition is taken
  - Compare with priorities in SyncCharts
- FSMs may be nondeterminate
  - Can mark transitions to permit nondeterminism



This example is a hierarchical FSM showing a thermostat with a nondeterminate fault mode.

Conclusion

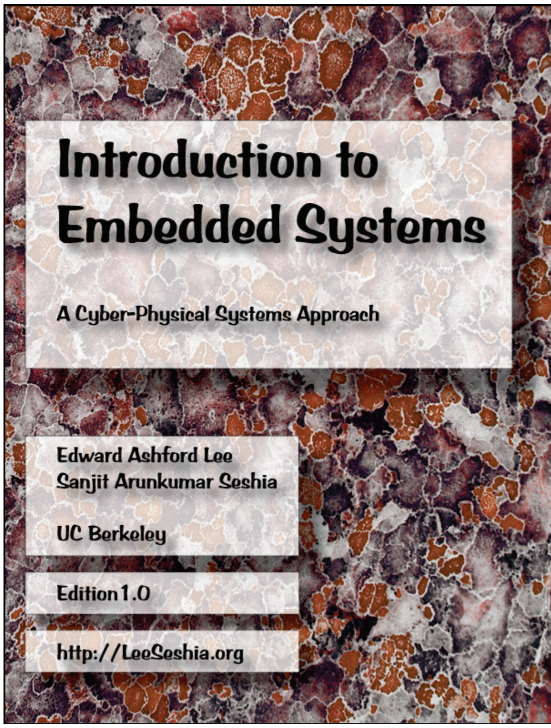
Modal models (in Ptolemy II, Statecharts, SyncCharts, Argos, etc.) provide a hierarchical mixture of imperative logic and declarative composition.

Humans are very capable of reasoning both *imperatively* (algorithms, recipes, etc.) and *declaratively* (equations, synchronous composition, etc.). We use these reasoning tools for *different* (complementary) tasks.

Models that support both will eventually replace models that provide only one or the other.

Acknowledgments

- Contributors to the modal model mechanisms in Ptolemy II:
  - Thomas Huining Feng
  - Xiaojun Liu
  - Haiyang Zheng
- Graphical editor in Vergil for state machines:
  - Stephen Neuendorffer
  - Hideo John Reekie
- Semantics of modal time:
  - Joern Janneck
  - Stavros Tripakis
- Online interactive examples and Ptolemy II infrastructure:
  - Christopher Brooks
- Other:
  - David Hermann & Zoltan Kemenczy (from RIM): transition refinements
  - Jie Liu: hybrid systems
  - Ye Zhou: modal dataflow models



New Text very relevant to this conference:

Lee & Seshia:
Introduction to Embedded Systems
- A Cyber-Physical Systems Approach

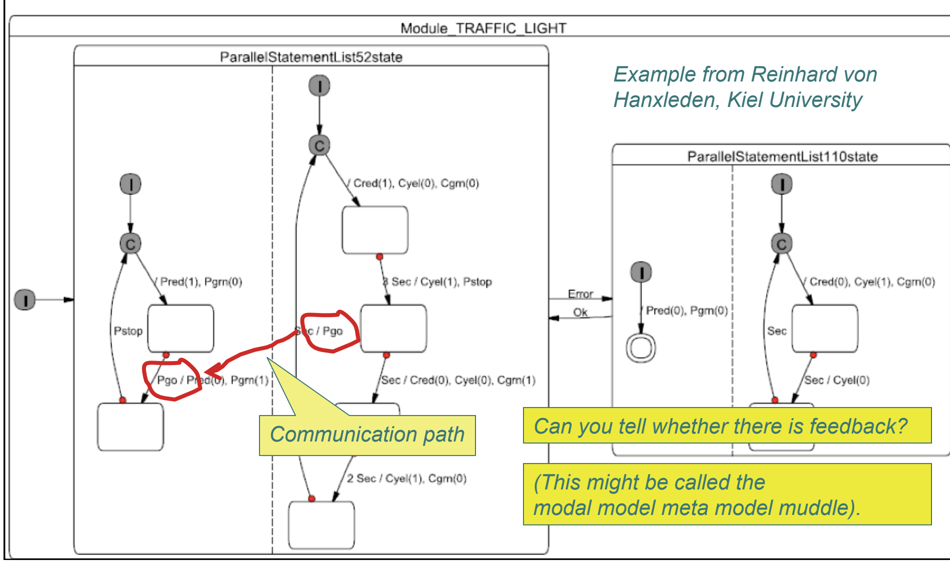
Electronic edition is available for free here:

http://LeeSeshia.org/

This book has a strong theme of model-based design of embedded systems.

Syntax of AND States

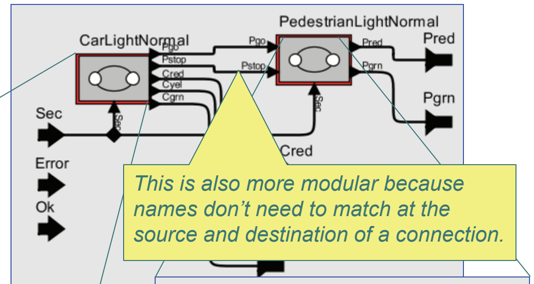
In Statecharts, communication between concurrent state machines is specified by name matching.

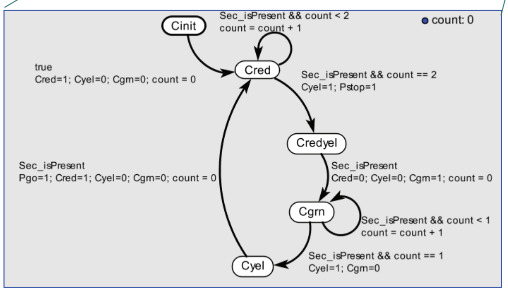


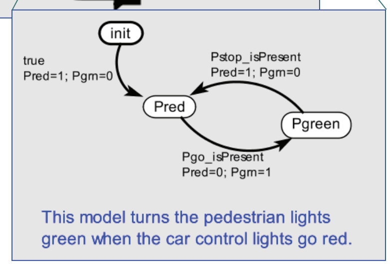
Syntax of AND States

In Ptolemy II, communication between concurrent state machines is explicit.

Now can you tell whether there is feedback?







Lee, Berkeley 35