Semantics of Modal Models in Ptolemy II

Edward A. Lee
Stavros Tripakis
UC Berkeley

Ninth Biennial Ptolemy Miniconference
February 16, 2011
What are modal models?

- Modal models = hierarchical models mixing FSMs (Finite State Machines) and other models

State refinement
Example: Hybrid System

Finite State Machine

- guard: abs(Force) > Stickiness
- set: Separate.p1 = P1;
- Separate.p2 = P2;
- Separate.v1 = v1;
- Separate.v2 = v1

- guard: touched_isPresent && (V1-V2) > 0.0
- set: Together.p = P1;
- Together.v = (V1+V2)/2.0;
- Together.stickiness = 10.0

Continuous-time model

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

Continuous-time model

- V1 and V2 are velocities, and P1 and P2 are positions of the two masses.
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
Ptolemy II goes one step further

**Arbitrary** combinations of FSMs with other domains
How to give meaning to modal models?

- Not always trivial:

What happens to the events produced by the discrete clock while system is at mode “irregular”?
How to give meaning to modal models?

- **Approach 1:**
  - Give a meaning to every possible combination of models:
    - Hierarchical state machines (Statecharts, UML, …)
    - Timed automata (timed models within state machines)
    - Hybrid automata (continuous models within state machines)
    - Mode automata (synchronous/reactive within state machines)
    - …
  - Scalable?
How to give meaning to modal models?

- Approach 2 [Ptolemy]:
  - Modular semantics
    - semantics of composite blocks = function of semantics of sub-blocks
  - Compositionality
  - Heterogeneity
This talk

- A formal semantics for Ptolemy
  - operational semantics
    - close enough to the Java implementation to be faithful
    - but not too close (fits in a few pages)

- A formal semantics for modal models
A FORMAL SEMANTICS FOR PTOLEMY
Abstract semantics

- Actor = State Machine
- Actor = Inputs + Outputs + States + Initial state + Fire + Postfire

- Fire = output function: produces outputs given current inputs + state
  \[ F : S \times I \rightarrow O \]

- Postfire = transition function: updates state given current inputs + state
  \[ P : S \times I \rightarrow S \]
Implemented as Java interfaces

**Interface “Executable”**

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> fire()</td>
</tr>
<tr>
<td>Fire the actor.</td>
</tr>
<tr>
<td><strong>boolean</strong> isFireFunctional()</td>
</tr>
<tr>
<td>Return true if this executable does not change state in either the prefire() or the fire() method.</td>
</tr>
<tr>
<td><strong>boolean</strong> isStrict()</td>
</tr>
<tr>
<td>Return true if this executable is strict, meaning all inputs must be known before iteration.</td>
</tr>
<tr>
<td><strong>int</strong> iterate(int count)</td>
</tr>
<tr>
<td>Invoke a specified number of iterations of the actor.</td>
</tr>
<tr>
<td><strong>boolean</strong> postfire()</td>
</tr>
<tr>
<td>This method should be invoked once per iteration, after the last invocation of fire() in that iteration.</td>
</tr>
<tr>
<td><strong>boolean</strong> prefire()</td>
</tr>
<tr>
<td>This method should be invoked prior to each invocation of fire().</td>
</tr>
<tr>
<td><strong>void</strong> stop()</td>
</tr>
<tr>
<td>Request that execution of this Executable stop as soon as possible.</td>
</tr>
<tr>
<td><strong>void</strong> stopFire()</td>
</tr>
<tr>
<td>Request that execution of the current iteration complete.</td>
</tr>
<tr>
<td><strong>void</strong> terminate()</td>
</tr>
<tr>
<td>Terminate any currently executing model with extreme prejudice.</td>
</tr>
</tbody>
</table>
Examples

**Single state ("stateless").**

\[ P : \text{trivial (state never changes)}. \]

\[ F : \text{out} := \text{in} \times 1.5 \]

**State: the current value**

\[ F : \text{out} := \text{state} \]

\[ P : \text{state} := \text{input} \]
Behaviors – untimed

- Set of untimed traces:

\[ \begin{align*}
S_0 \xrightarrow{x_0, y_0} S_1 \xrightarrow{x_1, y_1} S_2 \xrightarrow{x_2, y_2} \ldots 
\end{align*} \]

- such that for all \( i \):

\[ \begin{align*}
y_i &= F(s_i, x_i) \\
s_{i+1} &= P(s_i, x_i)
\end{align*} \]

\[ \begin{align*}
F : S \times I &\to O \\
P : S \times I &\to S
\end{align*} \]
What about “timed” actors?

States include special *timer* variables:

- Set to some positive value, “expire” when they reach 0, can be “frozen” and “resumed”, …
Behaviors – timed

- Set of timed traces:

\[ S_0 \xrightarrow{x_0,y_0,d_0} S_1 \xrightarrow{x_1,y_1,d_1} S_2 \xrightarrow{x_2,y_2,d_2} \ldots \]

- such that for all i:

\[
\begin{align*}
y_i &= F(s_i, x_i) \\
s_{i+1} &= P(s_i - d_i, x_i) \\
d_i &\leq \min\{\nu \mid \nu \text{ is the value of a timer in } s_i\}
\end{align*}
\]
What about hierarchy?

How to give semantics to a hierarchical model?
i.e.,

How to give semantics to a composite actor?

A NonStrictDelay actor "breaks" a feedback loop in a SR model.
Directors = composition operators

- Given a composite actor with a set of subactors $A_1, A_2, \ldots$, with fire & postfire functions $F_1/P_1, F_2/P_2, \ldots$

- … its director defines a new pair of fire & postfire functions $F$ and $P$.

- $F$ and $P$ define a new, composite actor $A$.

- $A$ can be used like an atomic actor (black-box).
Example: Synchronous/Reactive (SR)

To compute the composite \( F \), director solves a **fixpoint**:
- Keep on evaluating \( F_i \)'s until the values of all signals stabilize (this includes output signals in particular)
- State remains unchanged during computation of the fixpoint!

\[ \text{c.f. separation of fire and postfire} \]

To compute the composite \( P \), just execute all \( P_i \)'s
Example: Synchronous Data Flow (SDF)

In each firing, actors consume a fixed number of tokens from the input streams, and produce a fixed number of tokens on the output streams.

- **SDF Director computes periodic schedule**, e.g., A,A,A,B,B
- **Composite fire() fires all internal actors according to schedule**
MODAL MODEL SEMANTICS
Giving semantics to modal models

Goal:
define \( F_1, P_1 \) functions for the modal model

\( F_2, P_2 \) functions already defined for this refinement

\( F_c, P_c \) functions already defined for the "controller" automaton

\( F_1, P_1 \) 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 paper “Modal Models in Ptolemy” [EOOLT 2010]
Timed modal model example

- Mode transitions triggered at times 0, 2.5, 5, 7.5, etc.
- Events with value 1 produced at (local times) 0, 1, 2, 3, etc.
- First regular event generated at (global time) 0, then transition is immediately taken.
Conclusions, ongoing work and future challenges

- Modular formal semantics for Ptolemy II
  - Directors = composition operators over state machines

- Semantics worked out for modal models [EOOLT 2010]
  - Currently extending it to other domains: Synchronous-Reactive, SDF, Discrete-Event, Continuous-Time, …

- Meta-model to describe semantics?
Thank you

- Questions?