Ptera: An Event-Oriented Model of Computation for Heterogeneous Systems

Thomas Huining Feng
Oracle Corp.

Edward A. Lee
EECS, UC Berkeley

Lee W. Schruben
IEOR, UC Berkeley
Hierarchical Heterogeneous Modeling

Use the most appropriate modeling language for each component.

Compose those components to form more complex systems.
The Event-Oriented View

- Ptera (Ptolemy Event-Relationship Actor)
  - Based on event graphs [Schruben 1983]
- Visual representation
  - Nodes are events
  - Edges are scheduling relations

CarWash: single queue multiple servers

The CarWash model

- Compare
  - State diagram
  - UML activity diagram
  - Business process modeling
Execution with an Event Queue

\[
\begin{align*}
\delta: & \text{SimulationTime} \\
\text{Run} & \rightarrow \text{Enter} \quad \{ \text{Queue} = \text{Queue} + 1 \} \\
\text{Start} & \left\{ \begin{array}{l} 
\text{Servers} = \text{Servers} - 1; \\
\text{Queue} = \text{Queue} - 1 \end{array} \right. \\
\text{Leave} & \{ \text{Servers} = \text{Servers} + 1 \}
\end{align*}
\]

- \( \delta: 3.0 + 5.0 \cdot \text{random()} \)
- guard: \( \text{Servers} > 0 \)
- guard: \( \text{Queue} > 0 \)
- \( \delta: 5.0 + 20.0 \cdot \text{random()} \)

**Interarrival time**

**Service time**

- \( S = 3 \)
- \( Q = 0 \)
- \( S = 2 \)
- \( Q = 0 \)
- \( S = 2 \)
- \( Q = 1 \)
- \( S = 1 \)
- \( Q = 0 \)
- \( S = 2 \)
- \( Q = 0 \)
Simulation

\[
IAT = 3.0 + 5.0 \times \text{random()}
\]

\[
IAT = 1.0 + 5.0 \times \text{random()}
\]
Model Hierarchy: The Ptera Approach

- A submodel is itself a model
  - No difference in syntax
  - Conceptually equipped with an isolated event queue
  - A global notion of model time

- Implication: events (or tasks) are no longer instantaneous
  - Start of an event causes start of its submodel
  - End of the submodel causes end of the event
Communication via Ports

Event processing conditions
1. Scheduled time is reached, or
2. Tokens received at one or more triggering ports

Inputs not triggering any event are ignored.

Outputs can be sent in actions.
Goal: reusable, robust and flexible design
Choose DE at top level for
- Concurrency
- Concern separation
- Encapsulation
- Fixpoint semantics
- Out-of-order execution
- Distributed execution
Choose Ptera to model a random process
• No need to depend on predefined actors
• Easy to control the exact behavior
• Totally sequential (but concurrency may be possible)

Some predefined actors can be designed in this way
• Source actors
• Math actors
• Time delay actors
• Flow control actors
Hierarchical Multimodeling

Use FSM to capture two modes
- Easy to understand
- Easy to model check, debug, convert into other languages, ...

Submodel firing conditions
- The submodel itself requests (not in this case), or
- Input is received at a port, or
- The event containing the submodel is processed
A Ptera model similar to the standalone version. Listen to the input port for car arrivals.

Indefinitely wait for change of car number.

Output both S and Q when either of them is changed.
Opportunities

• Composition with other MoCs
  (Especially, Ptides and continuous time)

• Formal analysis
  (Bound of event queue, simultaneous events, termination condition, model categorization, ...)

• Behavior-preserving concurrent and distributed execution

• Other application domains
  (Currently studied: statistical analysis, model transformation)

• Tool support
  (Debugging and testing, code generation)

• Design patterns
  (Currently studied: Input, Output, LoopForCount, ParallelTasks, SingleQueueMultipleServers)