EADS

EADS INNOVATION WORKS

Systems Engineering

Execution of UML State Machines Using Modelica

Wladimir Schamai (EADS Innovation Works, Germany)
Uwe Pohlmann (University of Paderborn, Germany)
Peter Fritzson (Linköping University, Sweden)
Chris Paredis (Georgia Institute of Technology, USA)
Philipp Helle (EADS Innovation Works, UK)
Carsten Strobel (EADS Innovation Works, Germany)

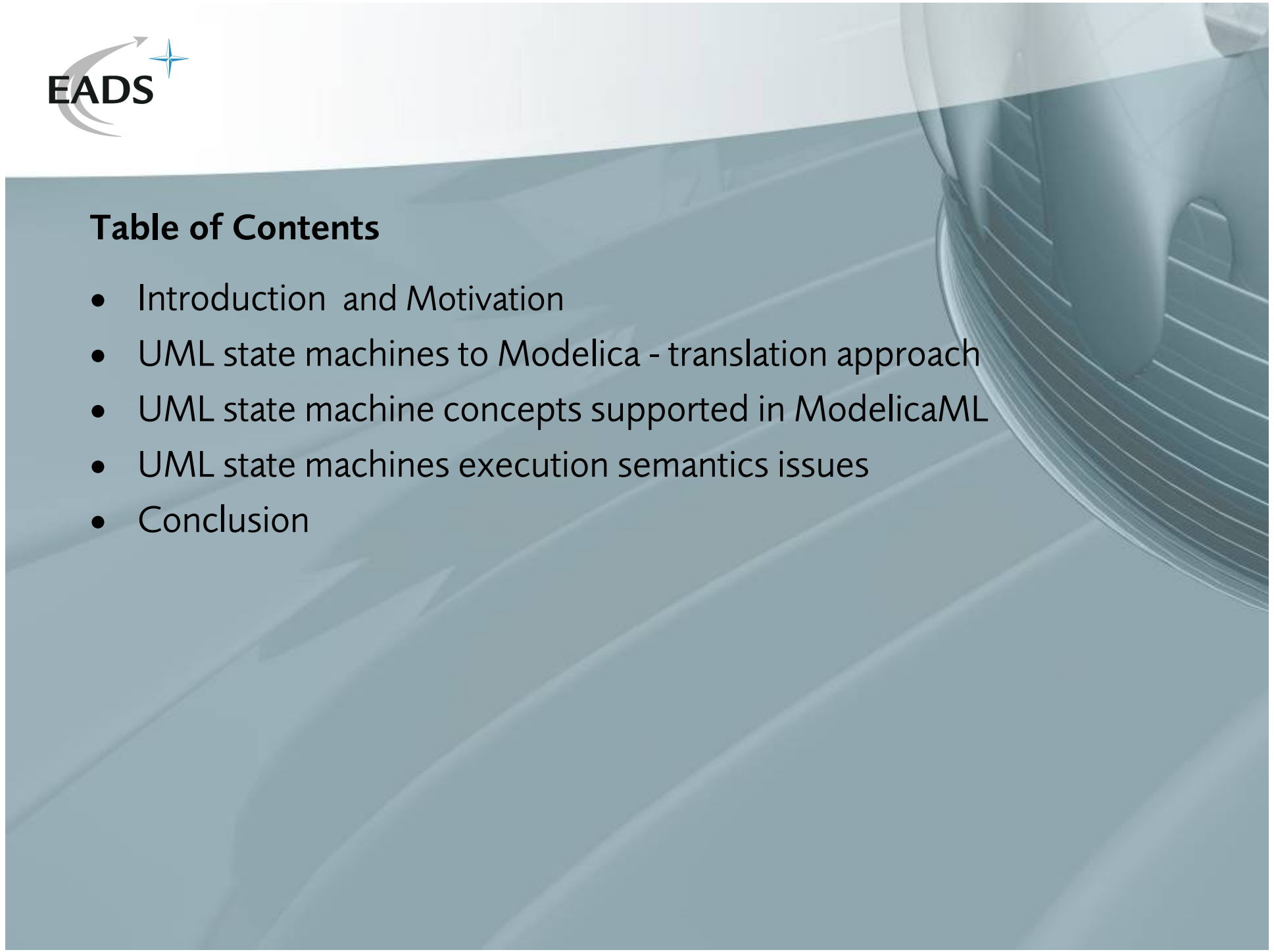
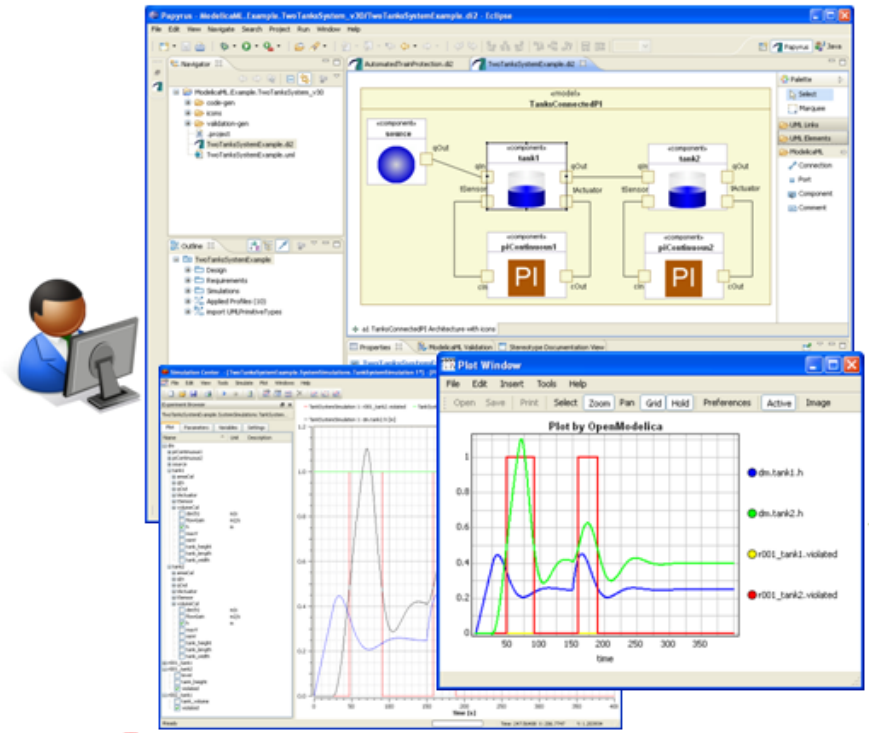


Table of Contents

- Introduction and Motivation
- UML state machines to Modelica - translation approach
- UML state machine concepts supported in ModelicaML
- UML state machines execution semantics issues
- Conclusion

Introduction: ModelicaML Concept

1 System Modeling with ModelicaML



2 Modelica Code Generation

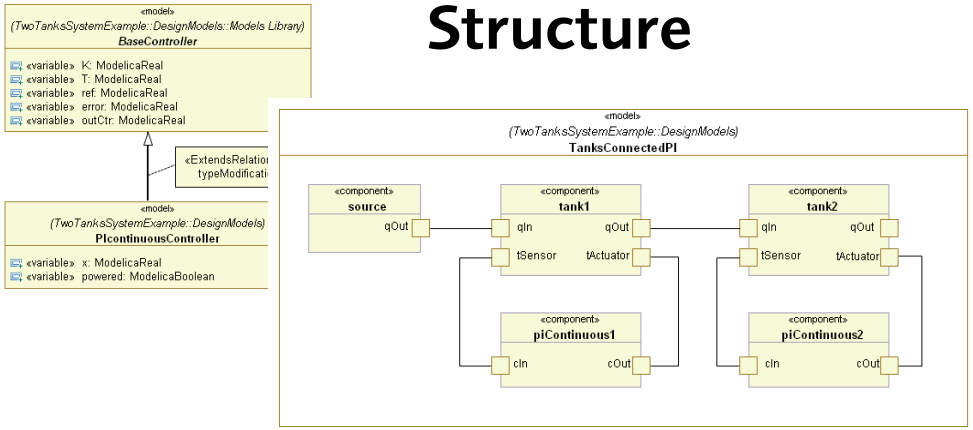
```
within TwoTanks;

function limitValue

  input Real pMin;
  input Real pMax;
  output Real pLim;
  input Real p;

  algorithm
  // code generated from the Activity "algorithm (diagram)"
  // Activity.name: "algorithm (diagram)"
    if p < pMin then
      pLim := pMin; // OpaqueAction.name: "pLim := pMin;"
    elseif p > pMax then
```

```
within DesignModels;

model TanksConnectedPI

  TwoTanksSystemExample.DesignModels.LiquidSource source(flowLevel = 0.02);
  TwoTanksSystemExample.DesignModels.Tank tank1;
  TwoTanksSystemExample.DesignModels.Tank tank2(tank_length = 1);
  TwoTanksSystemExample.DesignModels.PIcontinuousController piContinuous1(ref = 0.25);
  TwoTanksSystemExample.DesignModels.PIcontinuousController piContinuous2(ref = 0.4);

  equation
    connect(source.qOut, tank1.qIn);
    connect(piContinuous1.cIn, tank1.tSensor);
    connect(piContinuous1.cOut, tank1.tActuator);
    connect(tank1.qOut, tank2.qIn);
    connect(piContinuous2.cIn, tank2.tSensor);
    connect(piContinuous2.cOut, tank2.tActuator);

end TanksConnectedPI;
```

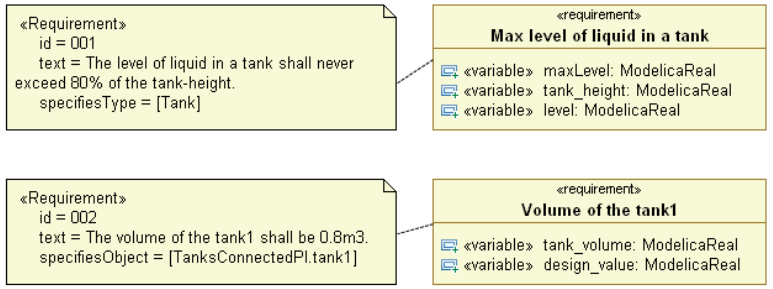
3 System Simulation with Modelica Tools

Introduction: ModelicaML Graphical Notation

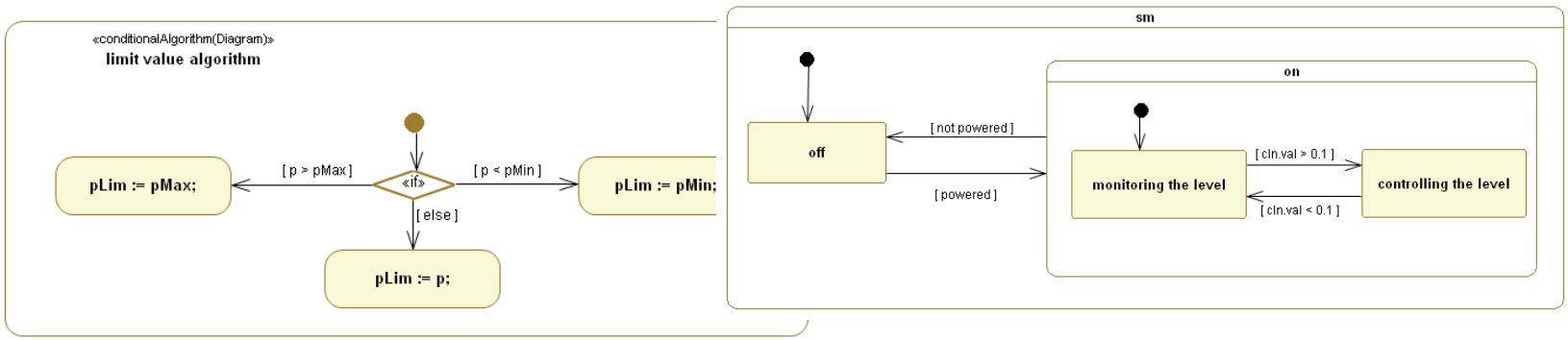
Structure



Requirements



Behavior



Introduction: Motivation

- UML state machines is a powerful formalism for describing system or components states and modes

- To which extent can UML state machines concepts be supported in ModelicaML?
- How to implement it?
  - Create a Modelica library or implement a code generator?
  - Use algorithmic code or equations?

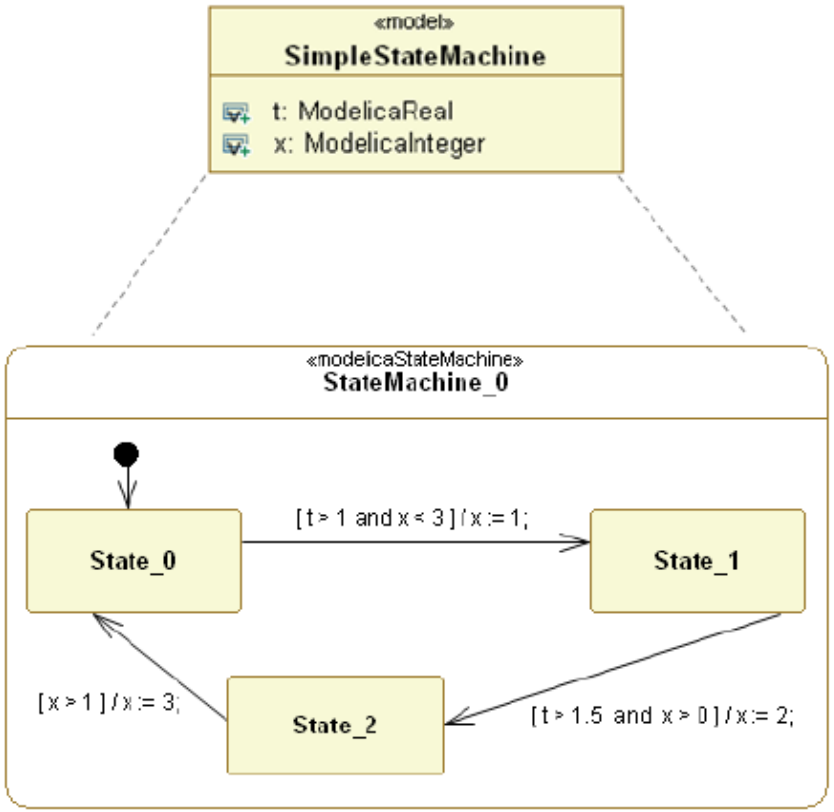
- This presentation focuses on the resolution of issues identified when translating UML state machines into executable Modelica code

Translation Approach

- Code generator approach was chosen instead library
  - States specification (i.e. in/outgoing transitions, entry, do, exit actions) cannot be predefined as a library component.

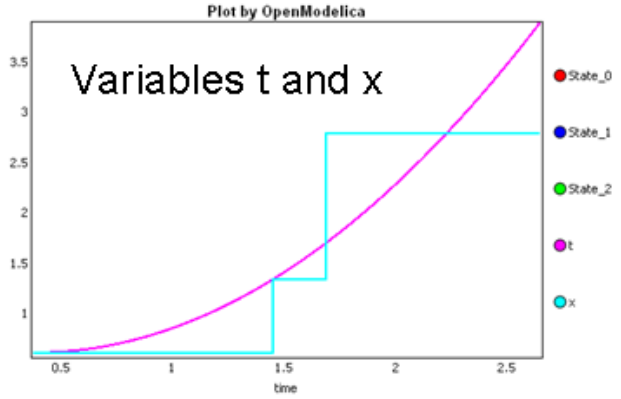
- ModelicaML code generator generates algorithmic Modelica code from ModelicaML state machines because:
  - The behavior of a state machine is always causal
  - For inter-level transitions, i.e. transitions which cross states hierarchy borders, the deactivation and activation of states and the execution sequence of associated actions (exit/entry action of states or state transitions effects) has to be performed in an explicitly defined order.

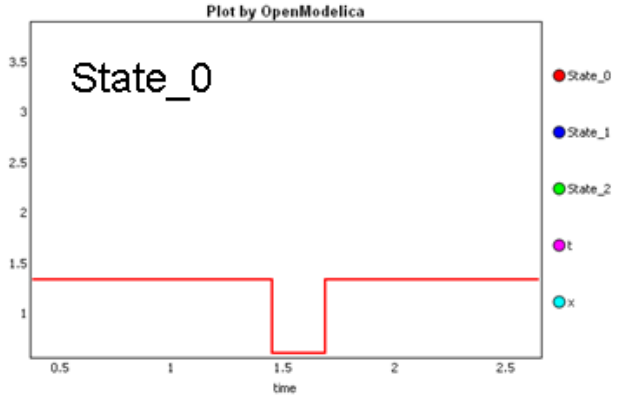
Simple Example (1/2)

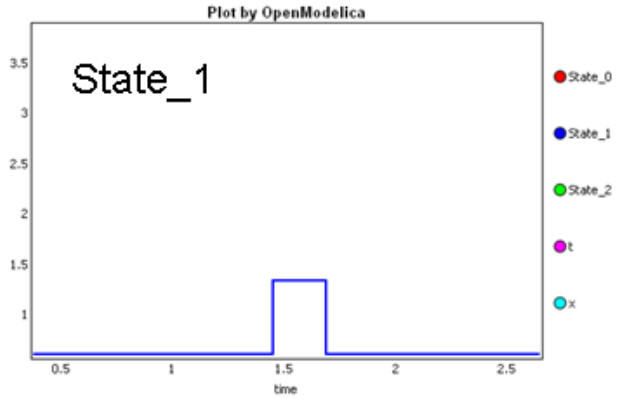


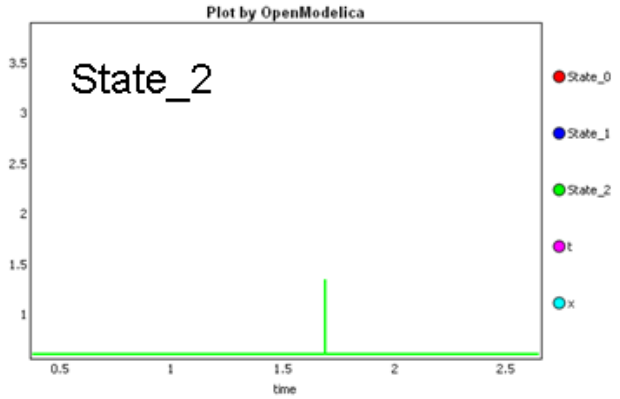

```
model SimpleStateMachine
  Boolean State_0 "State_0 representation";
  Boolean State_1 "State_1 representation";
  Boolean State_2 "State_2 representation";
  Integer x "discrete variable";
  Real t "continuous variable";
 equation //Code for continuous integration of t
    der(t)=time;

 algorithm //Code for StateMachine_0
  when initial() then
      State_0:=true "Activation of the initial state";
  end when;
  //Transition from State_0 to State_1
  if pre(State_0) and t > 1 and x < 3 then
      State_0:=false "Deactivation of state";
      x:=1 "Transistion effect";
      State_1:=true "Activation of state";
  //Transition from State_1 to State_2
  elseif pre(State_1) and t > 1.5 and x > 0 then
      State_1:=false "Deactivation of state";
      x:=2 "Transistion effect";
      State_2:=true "Activation of state";
  //Transition from State_2 to State_0
  elseif pre(State_2) and x > 1 then
      State_2:=false "Deactivation of state";
      x:=3 "Transistion effect";
      State_0:=true "Activation of state";
  end if;
end SimpleStateMachine;
```

Simple Example (2/2)





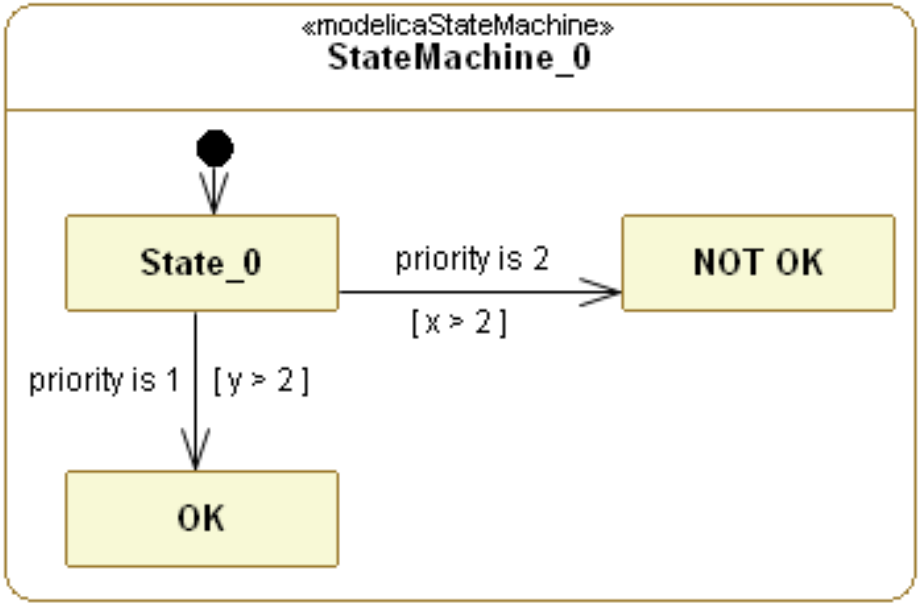




UML State Machine Concepts Supported in ModelicaML

- Hierarchical states modeling
  - Composite states
  - Sub-state machines (reusable for multiple states)
- Regions (orthogonal states)
- Pseudo states
  - initial , shallowHistory, join, fork, junction, choice, entryPoint, exitPoint, terminate
- Transition
  - Compound transitions (a transitions set from state to state through pseudo states)
  - Inter-level transition (transitions that cross hierarchy levels)
- Events
  - Change Events, Time Events, Signal Events
- State Actions
  - onEntry, Do, onExit

Issues with Conflicting Transitions



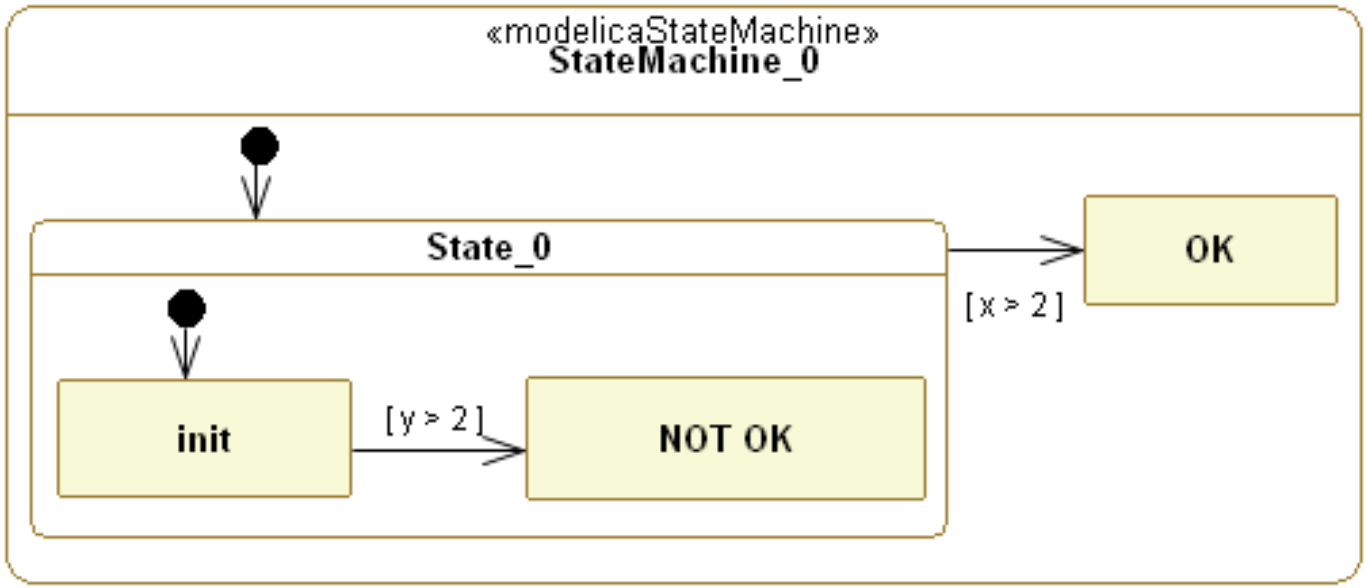
What happens when x and y are greater than 2 at the same time?

In ModelicaML: *Transition which the higher execution priority is taken (i.e. to state OK)*

Priority Order

1: target-> OK[x > 1]
2: target-> NOT OK[x > 2]

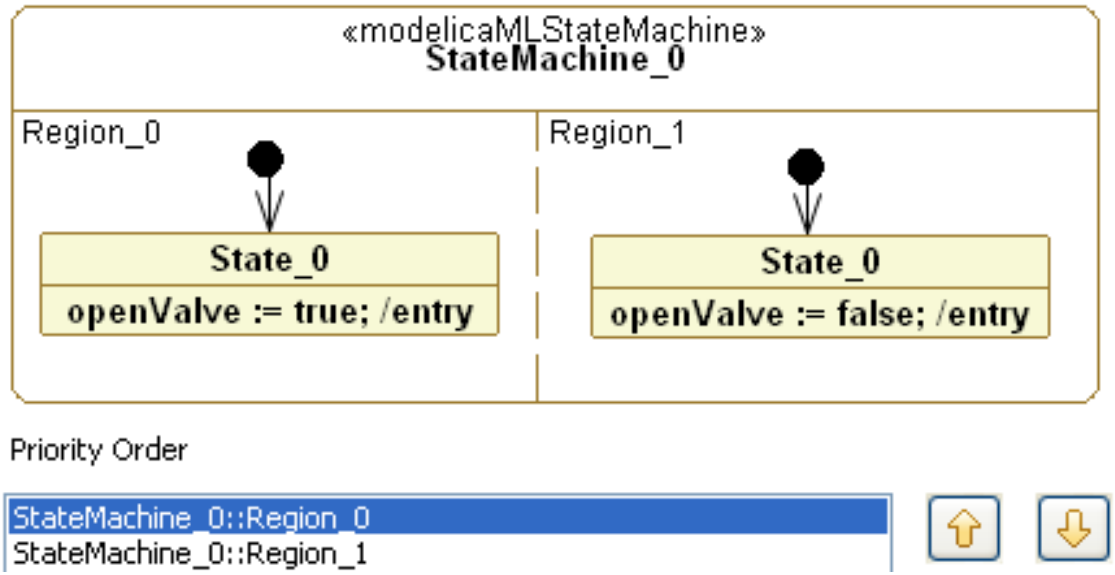
Priority Schema for Conflicting Transitions at Different State Hierarchy Levels



What happens when **x** and **y** are greater that **2** at the same time?

In ModelicaML: *Transition to state "OK" is taken because it is at higher hierarchy level*

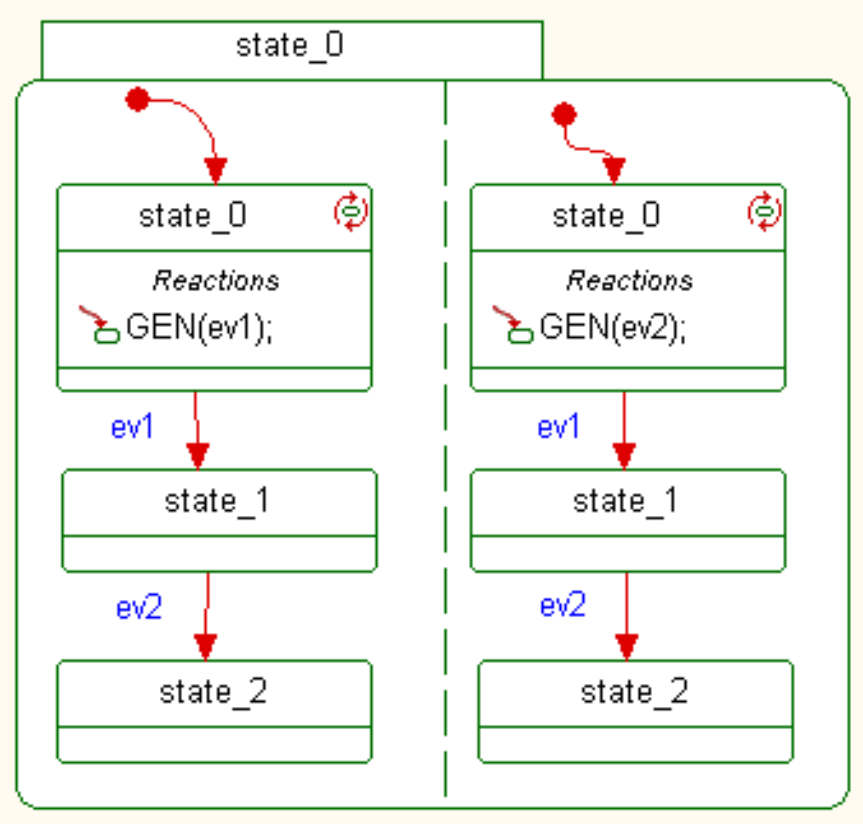
Issue with Concurrent Execution in Regions



What is openValve set to?

In ModelicaML: *To "false" because Region_0 was executed first and then Region_1 is execute.*

Issues With Concurrency When Using Event Queues (1/4)



-State machine enters the state_0 in both regions

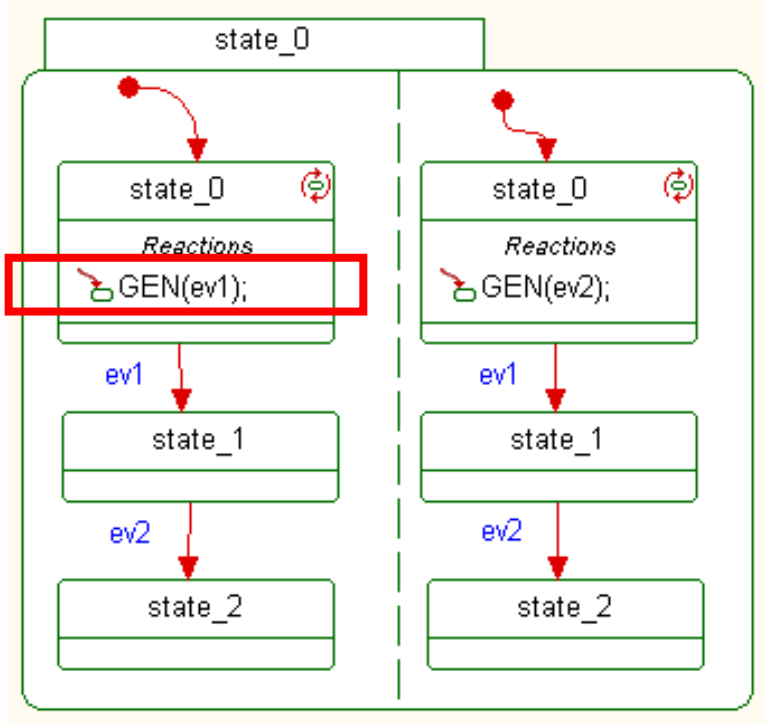
-On entry ev1 and ev2 are generated in state_0 in both regions

-Transitions to state_1 is performed in both regions

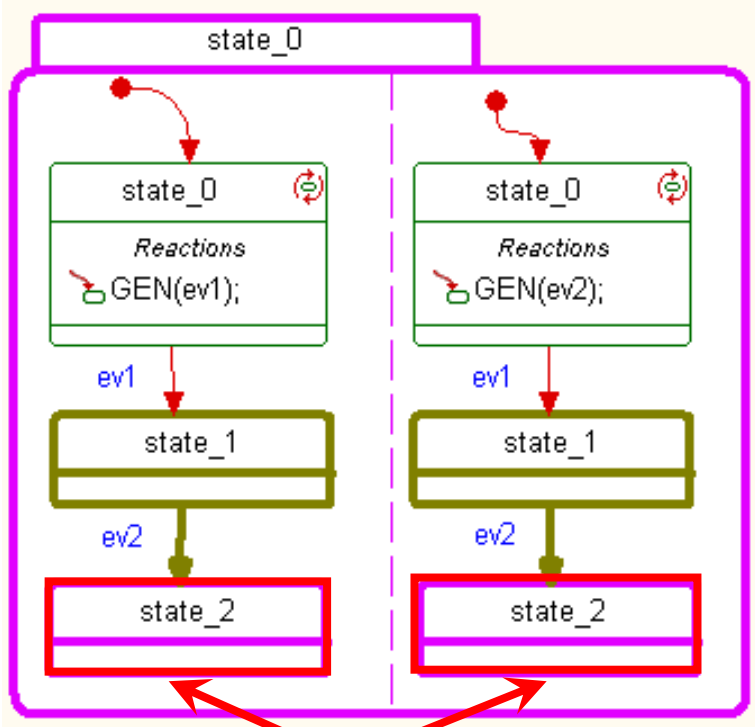
-...

What are the resulting active states configuration?

Issues With Concurrency When Using Event Queues (2/4)

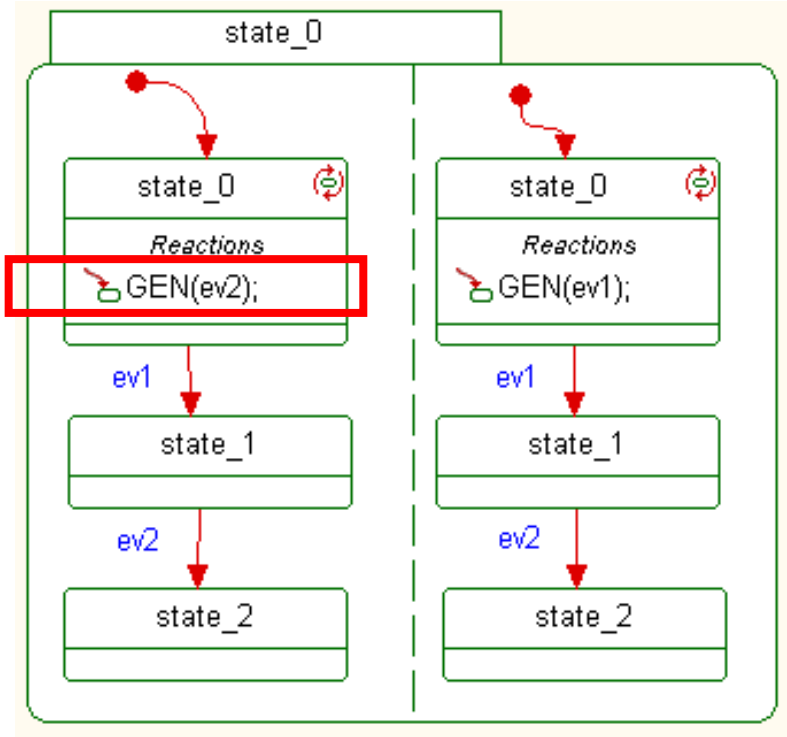


Simulation result in IBM Rhapsody

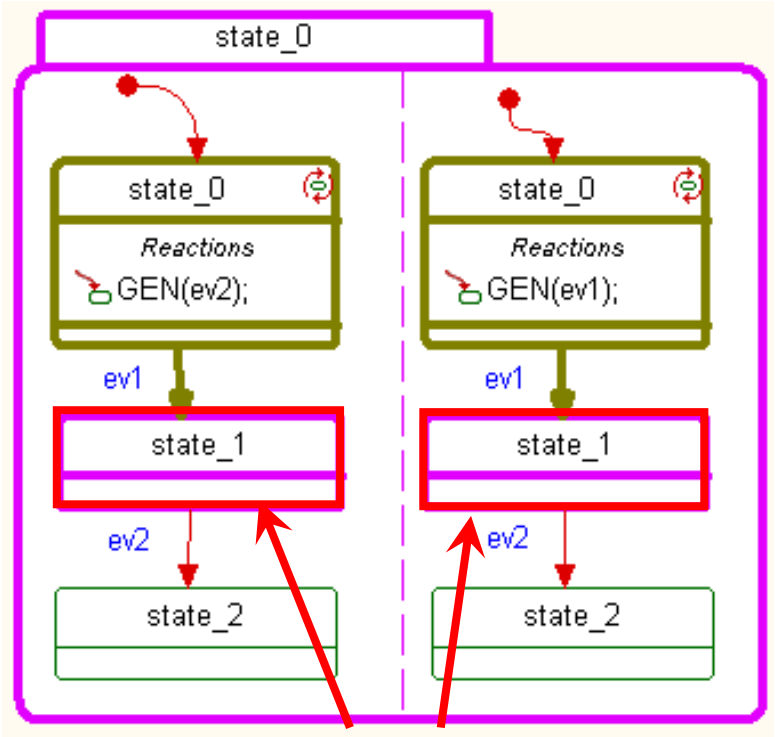


Active (resulting) states

Issues With Concurrency When Using Event Queues (3/4)

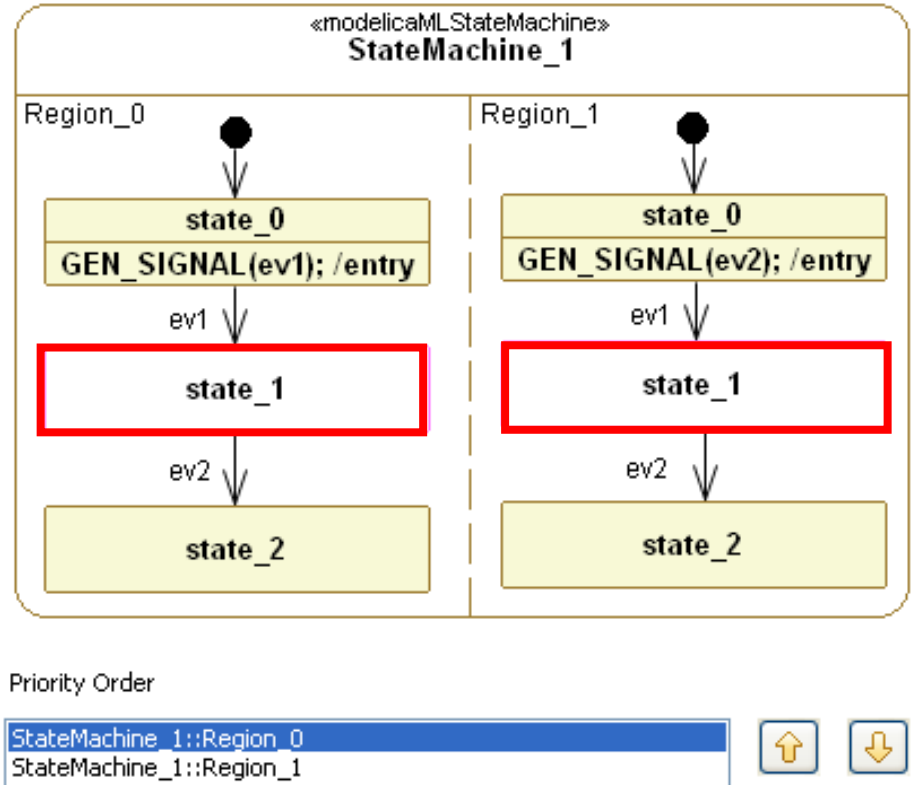


Simulation result in IBM Rhapsody



Active (resulting) states

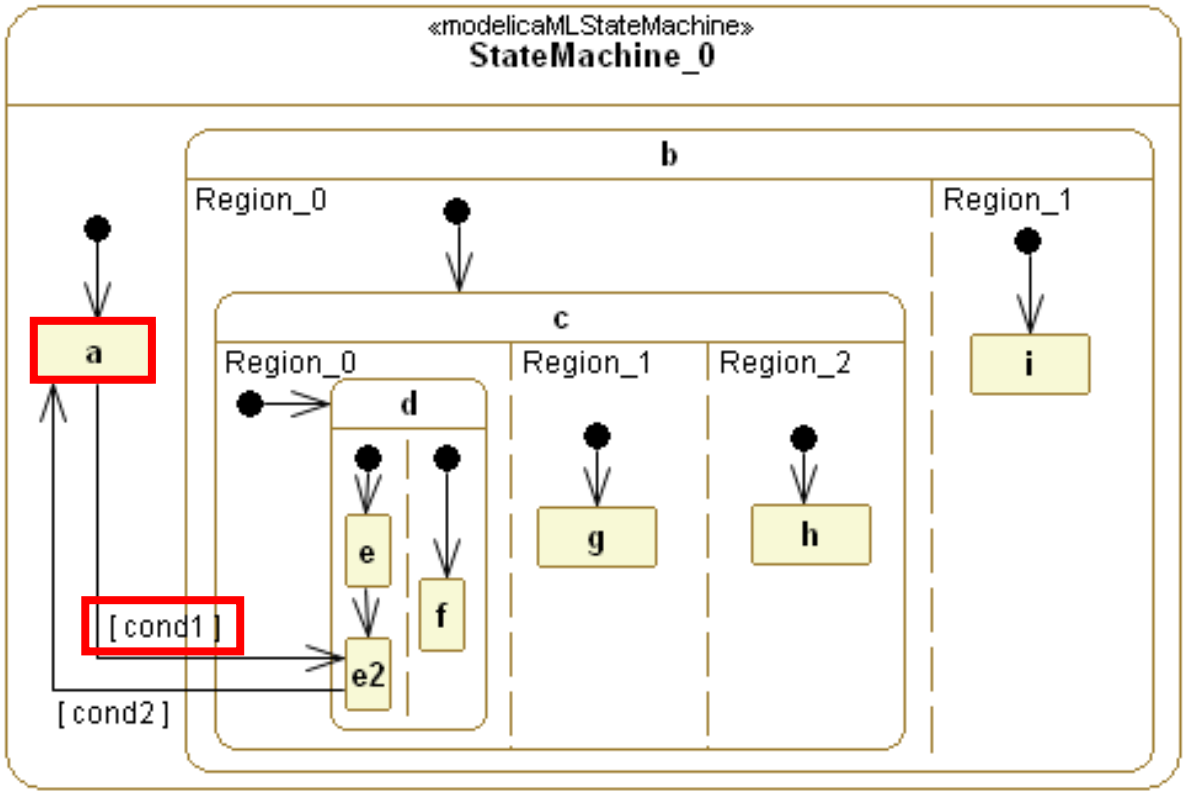
Issues With Concurrency When Using Event Queues (4/4)



What are the resulting states?

In ModelicaML: *Independent of whether ev1 is generated in Region_0 or in Region_1 the state machine ends up in state_1 in both regions because both events are generated when the state machines is in state_0 in both regions and are both consumed after the transition to state_1 is performed.*

Issues with Inter-Level Transitions

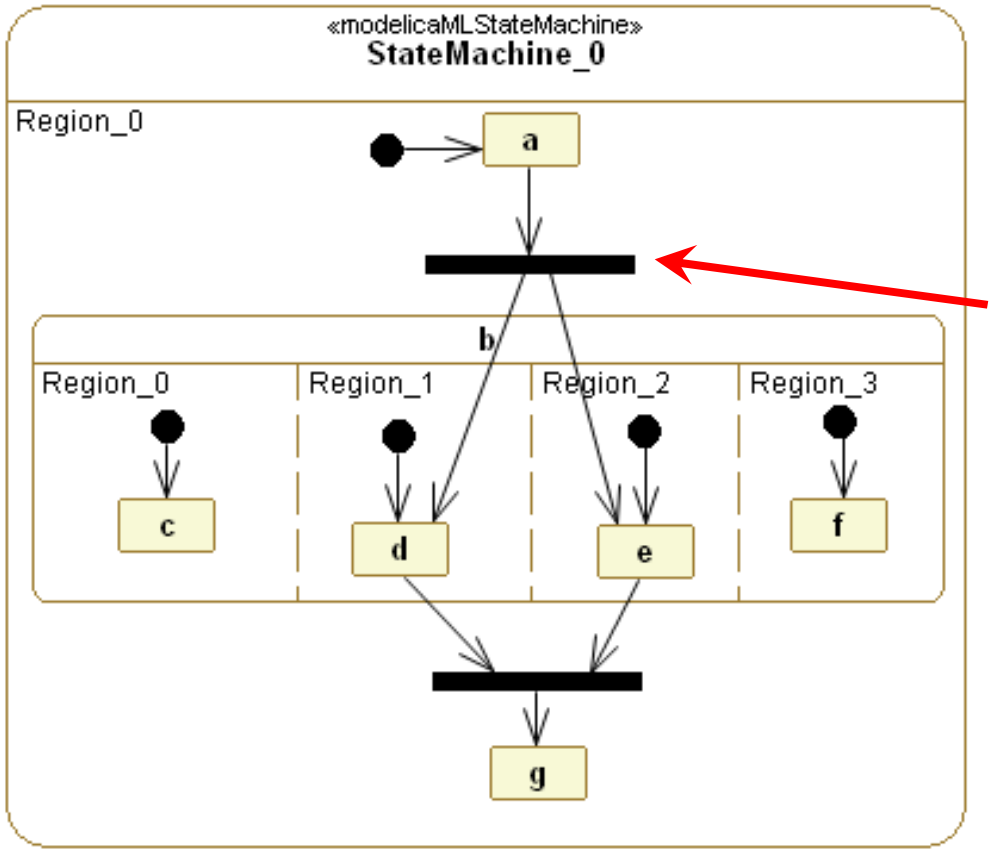


In which order are the states activated when cond1 becomes true?

b, c, d, e2, i, h, g, f
b, c, d, e2, i, g, h, f
b, c, d, e2, f, g, h, i
b, c, d, e2, f, h, g, i

In ModelicaML:
b, c, d, e2, f, g, h, i

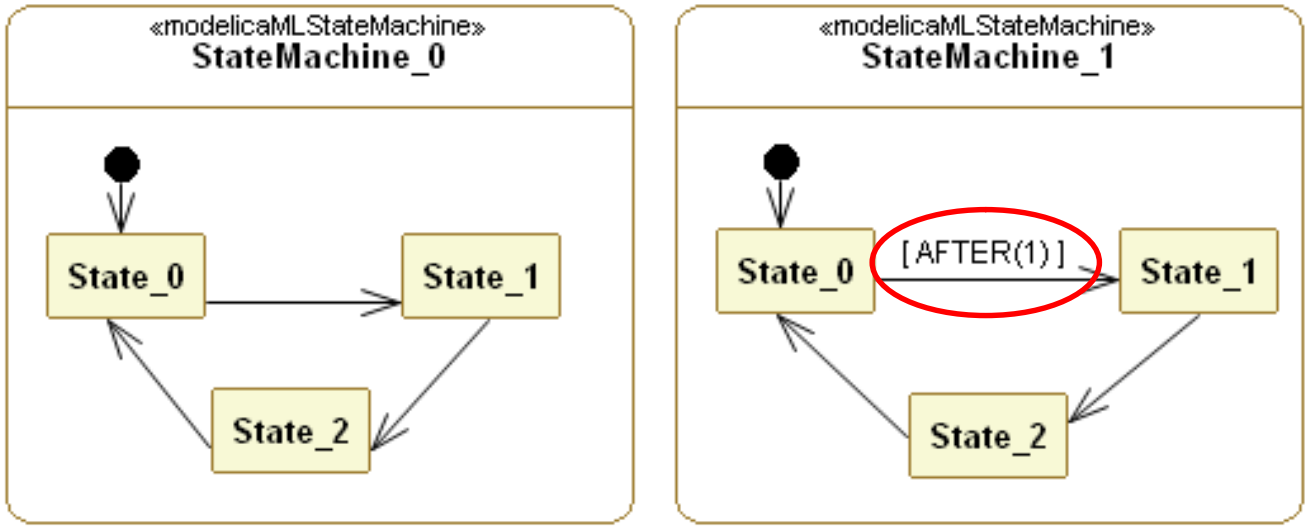
Issues with Fork and Join



In which sequence are states b, c, d, e, and f activated when the transitions (fork construct) from state a is executed?

In ModelicaML: *b*, *d* and *e* (based on the fork-outgoing transitions priority), *c* and *f* (based on their region priority)

Issues with Instantaneous States: Deadlocks (Infinite Looping)



Time-delayed transitions for breaking infinite looping at the same time instant.

Conclusion

- Using Modelica as execution language it is possible to support a comprehensive set of UML state machines specification

- Suggested improvement of the UML state machines specification
  - Priority for conflicting state-outgoing transitions
  - Priority for regions
- This enhancement of specification will clarify semantic and ensure
    - That state machine behavior is deterministic
    - That state machine behaves as intended by the modeler

Thank you for your attention!

EADS Innovation Works
Systems Engineering Team

Wladimir Schamai
Wladimir.Schamai@eads.net