ProDEVS
an Event-Driven Modeling and Simulation Tool for Hybrid Systems using State Diagrams

Le Hung Vu - Damien Foures – Vincent Albert

LAAS-CNRS / University of Toulouse

SIMUTOOL’15 - Athens, Greece
Agenda

• Introduction
• ProDEVS State Machine
• Discrete event simulation of continuous systems
• Discrete event simulation of hybrid systems
• M&S-based development approach and use case
• Conclusion and perspectives
Introduction
Objectives and motivation

• Object: Research platform for simulation, co-simulation and real-time simulation
• Scope: Discrete systems, continuous systems and hybrid systems
• Application: Cyber physical and embedded systems
• Objectives of research:
  – Study of new technology for simulation (asynchronous algorithms for numerical integration, concurrency, hardware (FPGA) computation)
  – Elaborate (executable) model-based design approach for heterogeneous systems.
Features

DEVS formalism
\[ M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle \]

UML StateMachine
Specialization

ProDEVS model specification

Abstract Simulator

Simulation Verification

ProDEVS \models \varphi_{sim}

M2M Transformations

SM_{TTS} \models \varphi_{SELT}

Trajectory Visualization and model animation


Technologies

- Java and OSGI (Open Service Gateway Initiative) with equinox
- JGraphX, JFreeChart
- Execution of compiled Java Code
- Serialization
- XML import/export
- FMI++ for cosimulation (work in progress)
ProDEVS State Machine
Syntax and semantic

• Typed Inputs, outputs and parameters
• States: periods in the life of the atomic component where it is expected some events to occur (input event or time event).
• A positive real value including 0 and infinity is given for each state to specify the time in which the system has to stay before triggering the output and the internal transition
• Transitions: oriented arcs define the relationships between states in case of time event (internal transition) and input event (external transition).
Example : GEN-BUF-PROC

\[ t = 0 : \text{event-list} = \{\text{GEN!job,2}\} : \text{q} = 0 \]
Example: GEN-BUF-PROC

t = 0 : event-list = \{GEN!job,2\} : q = 0

t = 2 : event-list = \{BUF!req,0;GEN!job,2\} : q = 1
Example : GEN-BUF-PROC

t = 0 : event-list = \{GEN!job,2\} : q = 0

t = 2 : event-list = \{BUF!req,0;GEN!job,2\} : q = 1

t = 2 : event-list = \{GEN!job,2;PROC!done,3\} : q = 0
Example : GEN-BUF-PROC

t = 0 : event-list = \{GEN!job,2\} : q = 0

\begin{align*}
t = 2 & : \text{event-list} = \{\text{BUF!req,0;GEN!job,2}\} : q = 1 \\
t = 2 & : \text{event-list} = \{\text{GEN!job,2;PROC!done,3}\} : q = 0 \\
t = 4 & : \text{event-list} = \{\text{PROC!done,1;GEN!job,2}\} : q = 1
\end{align*}
Example : GEN-BUF-PROC

\[
\begin{align*}
t = 0 : \text{event-list} &= \{\text{GEN!job,2}\} : q = 0 \\
t = 2 : \text{event-list} &= \{\text{BUF!req,0;GEN!job,2}\} : q = 1 \\
t = 2 : \text{event-list} &= \{\text{GEN!job,2;PROC!done,3}\} : q = 0 \\
t = 4 : \text{event-list} &= \{\text{PROC!done,1;GEN!job,2}\} : q = 1 \\
t = 5 : \text{event-list} &= \{\text{BUF!req,0;GEN!job,1}\} : q = 1
\end{align*}
\]
Discrete event simulation of continuous systems
Discretizing time or state?

- Classical approaches for ODE, DAE and PDE numerical integration are all based on time discretization. They use extrapolation means which are Taylor series approximations. They are synchronous algorithms.
- There is a family of state discretization based numerical integration techniques called Quantized State System (QSS). They are asynchronous algorithms.

Discretizing time

Given a model under the form $\dot{x}(t) = f(x(t), u(t))$ where $x$ is the state vector and $u$ the input vector.

Given $x_i$ the $i^{th}$ state trajectory in function of simulated time. The value of the trajectory in a given point is given by

$$x_i(t + h) = x_i(t) + \frac{dx_i(t)}{dt} \cdot h + \frac{d^2x_i(t)}{dt^2} \cdot \frac{h^2}{2!} + \cdots$$

Example: Forward Euler (FE)

$$x_i(t + h) = x_i(t) + \frac{dx_i(t)}{dt} \cdot h$$
Discretizing state (1/2)

Given a quantized state system \( \dot{q}(t) = f(q(t), u(t)) \)

Given a quantum \( \Delta Q \) such that
\[
\Delta Q = |q(t + \Delta t) - q(t)|
\]

\[
q_i(t + \Delta t) = q_i(t) + \dot{q}_i(t). \Delta t
\]
Discretizing state (2/2)

1. The required time for the solution of $q(t)$ to change by $\Delta Q$ is approximatively

$$\Delta t = \begin{cases} \frac{\Delta Q}{|\dot{q}|} & \text{if } \dot{q} \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

At this time, the next state will be

$$q(t_k + \Delta t) = q(t_k) + \text{sign}(\dot{q}(t_k)) \cdot \Delta Q$$

2. If $\dot{q}$ changes of value before $t_k + \Delta t$ then

$$q = q + \dot{q} \cdot e$$

$$\Delta t = \begin{cases} \frac{\Delta Q - |q - q_l|}{|\dot{q}|} & \text{if } \dot{q} \neq 0 \\ \infty & \text{otherwise} \end{cases}$$
ProDEVS 2\textsuperscript{nd} order ODE model

DC Motor equations

\[ L \frac{di}{dt} + R \cdot i + K_e \cdot \omega = u \]

\[ J \frac{d\omega}{dt} + f \cdot \omega = K_m \cdot i - C_r \]
Simulation results

QSS Simulation with ProDEVS

DASSL Simulation with OpenModelica
Discrete event simulation of hybrid systems
Discontinuities

- Sudden and unexpected change of the trajectory behavior during integration

```
block SampleBlock

...  

equation
    dx = 2x + 3u;

    when x > 10 then
        u = 2;
    end when;

end SampleBlock;
```

X > 10 is false at the integration step before $t_1$

X > 10 is true at the integration step after $t_1$
Example: Bouncing Ball
Example: DC-Motor PWM Controller
M&S-based development approach and use case
M&S-based development approach

- GUI
- ProDEVS
- Timed PetriNet .net + .c (.so)
- TINA-DYN

Bisimilar Model Transformation

- Exhaustive exploration and model checking
- Code generation
- ModelSim Synthesizable VHDL Code

- Synthesis
- FPGA

- Cosimulation

- Simulink
- OpenModelica
- C/C++
- Java

The FMI++ Library – Version 1.0 - v2015-02-11-17-29 - AIT Austrian Institute of Technology W. Muller and E. Widl. Linking FMI-based components with discrete event systems
Use Case: RobAFIS’14

Modeling and Simulation
- ProDEVS
- OpenModelica

Control Laws

PWM & Motors
- Simbad Java 3D

Rapid Prototyping and HIL simulation
- ZedBoard

Environment
- Control Laws + PWM
Conclusion and perspectives
Conclusion and perspectives

• Conclusion
  – Event-driven simulation tool using state diagram
  – Includes component libraries for continuous and hybrid systems
  – Discrete, continuous and hybrid models coexist in a same environment
  – Formal Methods and simulation
  – FPGA in the loop to boost the simulation performance

• Perspectives
  – Develop component libraries for physical domains
  – Integrate DS-DEVS or Rec-DEVS for FPGA Dynamic Reconfiguration
  – Develop event-driven and time-driven master algorithm for FMI++ cosimulation