Design and Development of a Dymola/Modelica Library for Discrete Event-oriented Systems using DEVS Methodology

Tamara Beltrame
Adviser: Prof. François E. Cellier
Responsible: Prof. Walter Gander

9th March 2006

Outline

Goal

- Development of a discrete-event systems library for Dymola.
- Enable simulation of continuous systems.

- **Implementation of a Modelica version of PowerDEVS.**

Motivation

Additional integration method for Dymola.

- Dymola is primarily designed for physical simulations.
- Physical systems are described by DAE's, need integration.
- QSS and the DEVS formalism are well suited for integration.
  - Idea: computers have to discretise.
  - Use state quantisation instead of time discretisation.
  - State variables evolve individually, no need to update them simultaneously.
  - A simulation of a QSS is numerically stable.
  - Formula for global error bound $\Rightarrow$ mathematical analysis.

In general: enable DEVS simulation within Dymola.

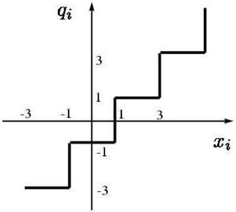
- For common discrete-event systems without integration.

Quantised State Systems (QSS)

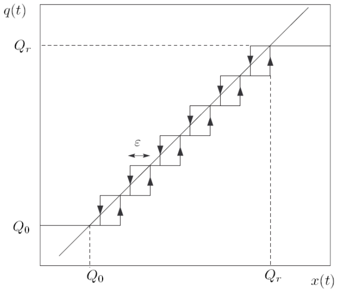
- QSS have piecewise constant input and output trajectories.
- Systems with piecewise constant trajectories can be simulated by the DEVS formalism
- QSS use a quantisation function to transform a continuous system into a system with piecewise constant input and output trajectories.
- Quantisation function is hysteretic in order to avoid illegitimate models.
  - Illegitimate models perform an infinite number of transitions in a finite interval of time.

Hysteretic Quantisation Function

- A quantisation function maps real numbers $x(t)$ into a discrete set of real values $q(t)$.
- Problem: $\dot{x}(t) = -sign(q(t))$



- A **hysteretic** quantisation function inhibits infinite oscillations within one time step.



Discretisation of a Continuous System

- Conventional continuous system: $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$
- Quantised continuous system: $\dot{\xi}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t), t)$

- Example: $\dot{x}(t) = -x(t) + 10\epsilon(t - 1.76)$

  Used quantisation function: $q(t) = \mathit{floor}(\xi(t))$

  $$\Rightarrow \quad \dot{\xi}(t) = -\mathit{floor}(\xi(t)) + 10\epsilon(t - 1.76)$$
  $$\Rightarrow \quad \dot{\xi}(t) = -q(t) + 10\epsilon(t - 1.76)$$

- $q(t)$ is a piecewise constant, linear or quadratic function.
  - QSS1 $\Rightarrow$ uses constant function.
  - QSS2 $\Rightarrow$ uses linear function.
  - QSS3 $\Rightarrow$ uses quadratic function.

The DEVS Formalism

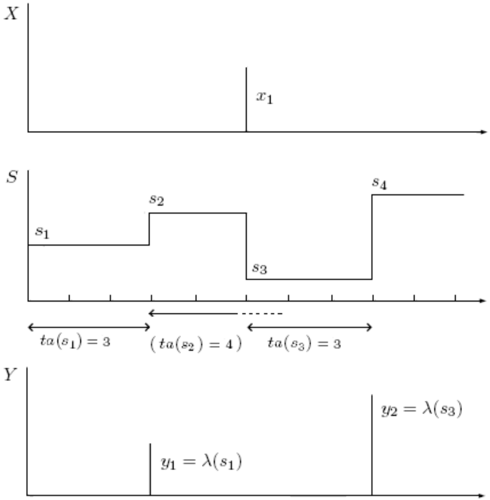
- Introduced by B. Zeigler in 1976.
- Discrete-event simulation methodology.
  Other discrete-event techniques: Petri nets, finite state machines, Markov chains, ...
- Particularity: DEVS models have infinite number of states
  $\Rightarrow$ useful for numerical integration.

Atomic Models

- Accepts an input trajectory (external events), generates an output trajectory.
- Definition: $M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$
  - X = set of inputs
  - S = set of possible states
  - Y = set of outputs
  - δ_{ext} = external transition
  - ta = time-advance function, often represented by σ
  - δ_{int} = internal transition
  - λ = output function

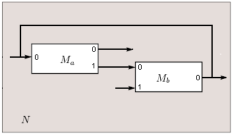
Atomic Models (cont.)

Example:



Coupled Models

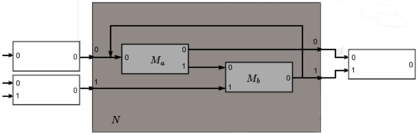
- DEVS is closed under coupling.



- Useful to split a complex model into simpler models.
- The dynamics of the coupled model N:
  1. Evaluate the atomic model d^* that is the next one to execute an internal transition. Let tn be the time when the transition has to take place.
  2. Advance the simulation time to $t = tn$ and let d^* execute the internal transition.
  3. Forward the output of d^* to all connected atomic models and let them execute their external transitions.

Hierarchic Models

- Reuse of coupled models as atomic models.



- The actual task of N is to wrap M_a and M_b, in order to make them look like as if they were one single model.
- The coupled model N features the same transitions as an atomic model, but the transitions of N depend on the transitions of its submodels.

The ModelicaDEVS Simulator

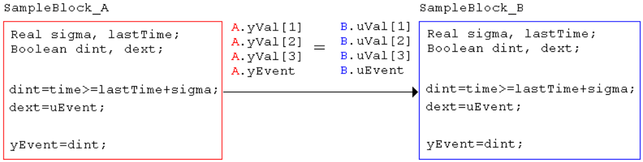
- Modelica models are described by equations.
  - Undirected data-flow: $x = y \Rightarrow$ either x or y has to be known.
    $2 + 4 = x \Rightarrow$ ok
  - Directed data-flow: $x := y \Rightarrow y$ has to be known.
    $2 + 4 := x \Rightarrow$ not ok
- Simultaneous equation evaluation $\Rightarrow$ parallel update of variables.
- Modelica is object oriented.

Atomic Models in ModelicaDEVS

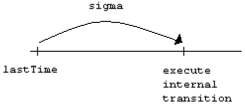
- ModelicaDEVS models have one or more input ports and one output port.
- ModelicaDEVS signals/events consist of the following values:
  - Coefficients of Taylor series up to second order of the current function value.
  - Boolean value. Indicates the creation of an event.
- Input event: `uVal[1]`, `uVal[2]`, `uVal[3]` and `uEvent`.
  Output event: `yVal[1]`, `yVal[2]`, `yVal[3]` and `yEvent`.
- Components have two Boolean variables `dint` and `dext`...
  - `dint=true` ⇒ execute internal transition.
  - `dext=true` ⇒ execute external transition.
- ... and two real-valued variables `lastTime` and `sigma`.
  - `lastTime` stores the time of the last event.
  - `sigma` stores the amount of time that has to elapse before the next internal transition takes place.

Coupled Models in ModelicaDEVS

- Communication between blocks:



- When block A executes its internal transition (dint=true) it sends an output to block B (yEvent=true).



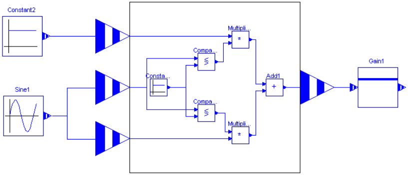
- When block B receives an event (uEvent=true) it executes its external transition.

Coupled Models in ModelicaDEVS (cont.)

- Benefit of the Dymola simulator:
  - Dynamics of coupled model still determined by its submodels.
  - Performs the same loop as defined by the DEVS formalism...
  - ... but the evaluation of $d*$ is done implicitly by Modelica's concept of simultaneous equation evaluation.

- **Coupled models are handled implicitly by the Dymola Simulator.**

Hierarchic Models in ModelicaDEVS

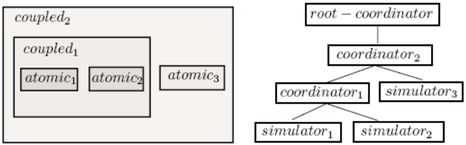


- A hierarchic model contains a component that consists of other components (submodels).
- Submodels just add a number of equations to the model equation "pool" $\Rightarrow$ no special treatment required.

- **Hierarchic models are handled implicitly by the Dymola Simulator.**

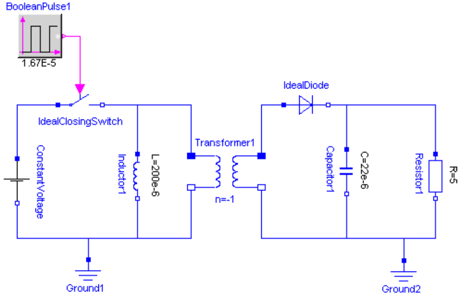
The PowerDEVS Simulator

- PowerDEVS is written in C++ ⇒ sequential variable updates.
- Hierarchical simulation scheme.



- Coordinators represent coupled models, simulators represent atomic models.
- Coordinators contain simulators or other coordinators.
- Coordinators control the interaction between their children.
  ⇒ Components on the same level do not communicate with each other, but only with their parent coordinator.

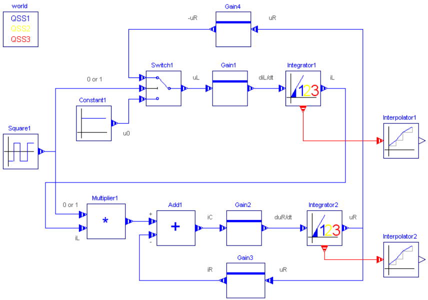
The Flyback Converter - Dymola



$$
\begin{aligned}
U_0 &= constant \\
0 &= \text{if } open_1 \text{ then } i_0 \text{ else } u_S \\
u_L &= L \cdot \frac{di_L}{dt} \\
i_C &= C \cdot \frac{du_R}{dt} \\
u_R &= R \cdot i_R \\
0 &= \text{if } open_2 \text{ then } i_D \text{ else } u_D \\
open_2 &= u_D < 0 \text{ and } i_D \leq 0 \\
u_T &= -u_L \\
i_T &= -i_D \\
i_0 &= i_L + i_T \\
i_D &= i_C + i_R \\
u_0 &= u_S + i_L \\
0 &= u_T + u_D + u_R
\end{aligned}
$$

The Flyback Converter - ModelicaDEVS/PowerDEVS

- ModelicaDEVS requires a block diagram representation.
  - ModelicaDEVS contains generic blocks, no electrical components
  - DEVS imposes certain data flow.
- Causalise equations by the Tarjan algorithm (x=y ⇒ x:=y).
- Model each (causalised) equation by a compound of blocks.



The Flyback Converter - Results

- Flyback converter simulated with Dymola, PowerDEVS and ModelicaDEVS (2ms of simulation time).
  - PowerDEVS needs 0.018s
  - Dymola (LSODAR) needs 0.062s, generates 738 result points
  - ModelicaDEVS (LSODAR, QSS3) needs 0.656s, generates 2164 result points
- PowerDEVS is faster than Dymola:
  - Dymola “suffers” from the simultaneous equation evaluation: PowerDEVS updates only the variables of the active component, Dymola updates **all** variables.
- Dymola is faster than ModelicaDEVS:
  - ModelicaDEVS generates a lot more result points than Dymola.
  - ModelicaDEVS models feature more variables (factor 3).

Summary

- Unfortunately, ModelicaDEVS is about 10 times slower than Dymola and about 40 times slower than PowerDEVS.
- Transformation of continuous systems described by equations into block diagrams is time consuming and sometimes problematic.

- ModelicaDEVS enables simulation according to the DEVS formalism within the Dymola environment.
- Possibility to combine standard Dymola simulation with DEVS.

Additional Example Hysteretic Quantisation Function

- Continuous system: $\dot{x} = -x + 0.5$, initial condition $x(0) = 2$
- Quantised system: $\dot{\xi} = -floor(\xi) + 0.5$

Dynamics

$$
\begin{array}{llll}
t = 0 & : \xi = 2 & \Rightarrow & \dot{\xi} = -1.5 \\
t = 0^{+} & : \xi = 1.999 & \Rightarrow & \dot{\xi} = -0.5 \\
t = 2 & : \xi = 1 & \Rightarrow & \dot{\xi} = -0.5 \\
t = 2^{+} & : \xi = 0.999 & \Rightarrow & \dot{\xi} = +0.5 \\
t = 2^{++} & : \xi = 1 & \Rightarrow & \dot{\xi} = -0.5 \\
t = 2^{+++} & : \xi = 0.999 & \Rightarrow & \dot{\xi} = +0.5
\end{array}
$$