Towards a Co-simulation Semantics of VDM-RT/Overture and 20-sim

Kenneth Lausdahl Joey W. Coleman Peter G. Larsen

Department of Engineering, Aarhus University, Denmark

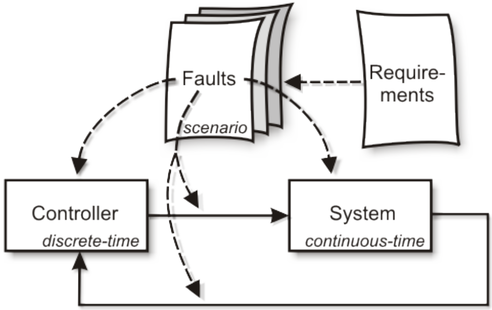
28 August 2012 / 10th International Workshop Overture/VDM

Outline

DESTECS
Design Support and Tooling for Embedded Control Software

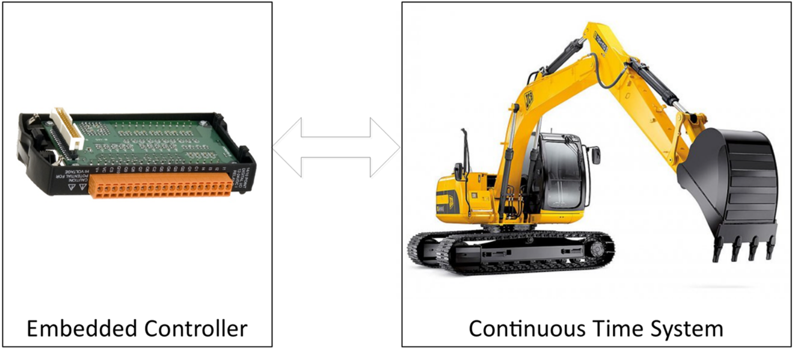
DESTECS is a European R&D project that aims to develop:

- Methods and Tools for modelling embedded control systems

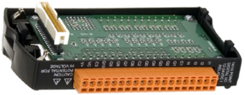


Embedded Control Systems

Embedded Control Systems



Embedded Control Systems



Embedded Controller

Continuous Time System

Co-Simulation Challenge

The main challenge in DESTECS is to:

- Combine the two tools:
  - Overture
  - 20-Sim

The general challenge is there by:

- Running the two models in parallel
- Synchronization of simulators

Co-Simulation Challenge

The main challenge in DESTECS is to:

- Combine the two tools:
  - Overture
  - 20-Sim

The general challenge is there by:

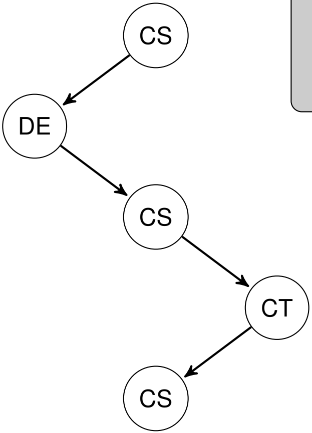
- Running the two models in parallel
- Synchronization of simulators

Objective

Define semantics for the co-simulation

- Define a toplevel semantics for co-simulation only
  - Separate VDM and co-simulation semantics
- Create a clean co-simulation interface
  - Allowing any compliant language

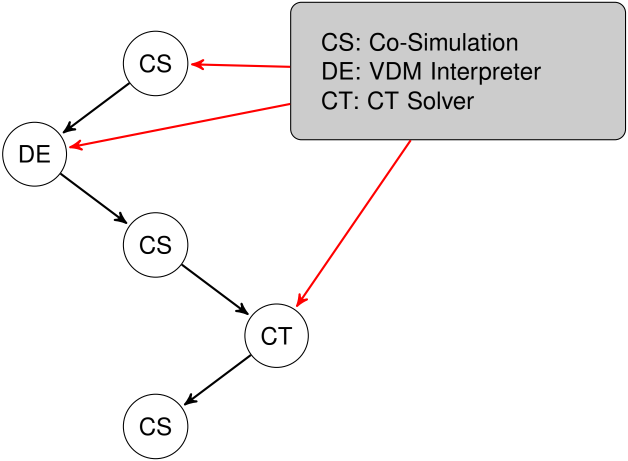
Co-Simulation
Abstract Overview



CS: Co-Simulation
DE: VDM Interpreter
CT: CT Solver

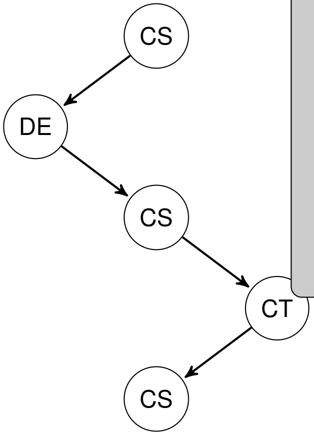
Co-Simulation

Abstract Overview



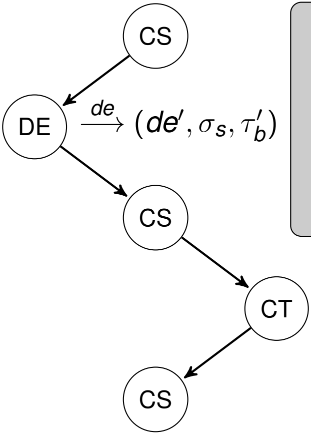
Co-Simulation
Abstract Overview

$$(de, \sigma_o, \tau_0, events) \xrightarrow{de}$$



- I Reads in values from σ_o
- II DE commits any pending values $\leq \tau_0$
- III Handles events
- IV Context switching and execution

Co-Simulation
Abstract Overview

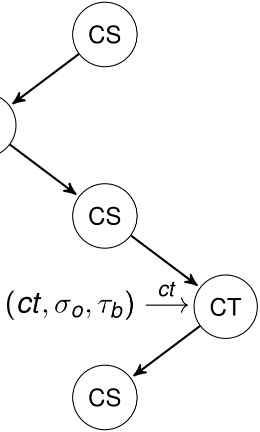


I Extracts values to σ_s

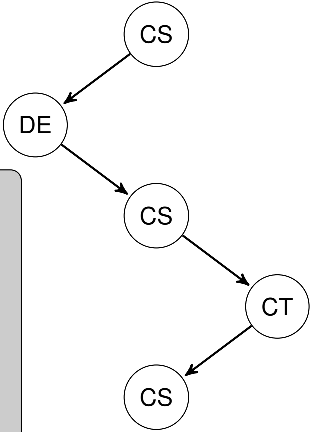
II Calculates a time bound $\tau_0 \leq \tau'_b$

Co-Simulation
Abstract Overview

- I Reads in values from σ_o
- II Calculates at most to τ_b



Co-Simulation
Abstract Overview



$\xrightarrow{ct} (ct', \sigma_s, \tau', events')$

- I Extracts values to σ_s
- II Returns time reached $\tau' \leq \tau_b$
- III Creates events $\tau' < \tau_b'$ $\Rightarrow$ $events' \neq \{\}$

Outline

Formal Semantics

The behaviour of a system can be described by transition relations. We are using Plotkin's Structural Operational Semantics as our framework.

$$C \longrightarrow C'$$

Where C is a configuration describing the complete system, including both program/control state and variable states.

Formal Semantics

Plotkin's SOS: Sequencing Rules

Plotkin's SOS defines transition relations by inference rules:

$$\boxed{\text{Sequencing}}\ \frac{(stm_1, \sigma) \longrightarrow (stm'_1, \sigma')}{(stm_1; stm_2, \sigma) \longrightarrow (stm'_1; stm_2, \sigma')}$$

$$\boxed{\text{Nil}}\ \frac{}{(\mathbf{nil}; stm_2, \sigma) \longrightarrow (stm_2, \sigma)}$$

Formal Semantics

Plotkin's SOS: While Rules

$$\boxed{\text{While true}}\ \frac{[\![B]\!]\sigma = \textbf{true}}{(\textit{mk-While}(B, \textit{stm}), \sigma) \longrightarrow (\textit{stm}; \textit{mk-While}(B, \textit{stm}), \sigma)}$$

$$\boxed{\text{While false}}\ \frac{[\![B]\!]\sigma = \textbf{false}}{(\textit{mk-While}(B, \textit{stm}), \sigma) \longrightarrow (\textbf{nil}, \sigma)}$$

Outline

1. Introduction
2. Semantic Framework
3. Co-Simulation Semantics

Co-Simulation
System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation
System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation
System Configuration

Type definition:

$$Config = DE \times \color{red}{CT} \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation

System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation
System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- DE: Discrete-event simulator
- CT: Continuous-time simulator
- Σ_o: Shared variables between the simulators

$$\Sigma_o = Id_v \xrightarrow{m} (SharedValue \times SysTag)$$

- $Time$: The current time
- $Time$: The time bound on the CT execution
- $Event$-**set**: Events generated by CT
- $SysTag$: The simulator that took the last step

Co-Simulation

System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation

System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation

System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-Simulation

System Configuration

Type definition:

$$Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event\text{-}\mathbf{set} \times SysTag$$

- *DE*: Discrete-event simulator
- *CT*: Continuous-time simulator
- Σ_o: Shared variables between the simulators
- *Time*: The current time
- *Time*: The time bound on the CT execution
- *Event*-**set**: Events generated by CT
- *SysTag*: The simulator that took the last step

Co-simulation Transition

The main co-simulation transition relations has the type:

$$\xrightarrow{cs} : \mathit{Config} \times \mathit{Config}$$

The transition $\xrightarrow{cs}$ is composed of the union of two rules Co-Sim DE Step and Co-Sim CT Step.

$$\langle DE \rangle \xrightarrow{cs} \langle CT \rangle \xrightarrow{cs} \langle DE \rangle \xrightarrow{cs} \langle CT \rangle \xrightarrow{cs} \cdots$$

Co-Simulation
DE Step

Co-Sim DE Step

$$\frac{(de, \sigma_o, \tau, events) \xrightarrow{de} (de', \sigma_s, \tau_b') \quad \sigma_o' = mergeStates(\sigma_o, \sigma_s)}{(de, ct, \sigma_o, \tau, \tau_b, events, \langle \mathrm{CT} \rangle) \xrightarrow{cs} (de', ct, \sigma_o', \tau, \tau_b', events, \langle \mathrm{DE} \rangle)}$$

Co-Simulation
CT Step

Co-Sim CT Step

$$\frac{(ct, \sigma_o, \tau_b) \xrightarrow{ct} (ct', \sigma_s, \tau', events') \quad \sigma'_o = mergeStates(\sigma_o, \sigma_s)}{(de, ct, \sigma_o, \tau, \tau_b, events, \langle \mathrm{DE} \rangle) \xrightarrow{cs} (de, ct', \sigma'_o, \tau', \tau_b, events', \langle \mathrm{CT} \rangle)}$$

CT Transition Relation

$$\xrightarrow{ct} : (CT \times \Sigma_o \times Time) \times (CT \times \Sigma_o \times Time \times Event\text{-}\mathbf{set})$$

CT Solver

Is a collection of equations, which runs to the time bound given; it is a differential equation solver.

DE Transition Relation

$$\xrightarrow{de}: (DE \times \Sigma_o \times Time \times Event\text{-}\mathbf{set}) \times (CT \times \Sigma_o \times Time)$$

DE Interpreter

The DE interpreter always runs ahead and commits pending changes when the CT solver catches up.

DE Rule

DE big step

$$\frac{\begin{array}{l} de^{1} = updateDEFromShared(de, \sigma_o) \\ de^{2} = commitPendingValuesAndUpdateTime(de^{1}, \tau) \\ de^{3} = createPeriodicAndEventThreads(de^{2}, events) \\ de^{4} = doContextSwitches(de^{3}) \\ de^{4} \xrightarrow{deexec} de^{5} \\ (\sigma_s, \tau_b) = extractValuesAndMinDurationFromDE(de^{5}) \end{array}}{(de, \sigma_o, \tau, events) \xrightarrow{de} (de^{5}, \sigma_s, \tau_b)}$$

Future Work

We are currently finishing the co-simulation and VDM-RT semantics but have left the following for future work:

- Complete the VDM-RT semantics
    - Exception handling
    - Object inheritance
- Reconcile the interpreter