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

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

Continuous Time System
Embedded Control Systems

Embedded Controller

Continuous Time System
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

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

CS
DE
CS
CT
CS

Lausdahl, Coleman, Larsen
Towards a Co-simulation Semantics of VDM-RT/Overture and 20-sim
Co-Simulation
Abstract Overview

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

DE
CS
CS
CS
CT
Co-Simulation
Abstract Overview

\((de, \sigma_0, \tau_0, \text{events}) \xrightarrow{de} \text{DE} \)

I Reads in values from \(\sigma_0\)

II DE commits any pending values \(\leq \tau_0\)

III Handles events

IV Context switching and execution

Lausdahl, Coleman, Larsen
Towards a Co-simulation Semantics of VDM-RT/Overture and 20-sim
Co-Simulation
Abstract Overview

**Introduction**
Semantic Framework
Co-Simulation Semantics

Lausdahl, Coleman, Larsen
Towards a Co-simulation Semantics of VDM-RT/Overture and 20-sim
Co-Simulation
Abstract Overview

I Reads in values from $\sigma_o$
II Calculates at most to $\tau_b$

$\left(ct, \sigma_o, \tau_b\right) \xrightarrow{ct} CT$

Lausdahl, Coleman, Larsen
Co-Simulation
Abstract Overview

1. Extracts values to $\sigma_s$
2. Returns time reached $\tau' \leq \tau_b$
3. Creates events $\tau' < \tau'_b \Rightarrow \text{events}' \neq \{\}$
The behaviour of a system can be described by transition relations. We are using Plotkin’s Structural Operational Semantics as our framework.

\[ C \xrightarrow{} C' \]

Where \( C \) is a configuration describing the complete system, including both program/control state and variable states.
Plotkin’s SOS defines transition relations by inference rules:

**Sequencing**

\[
(stm_1, \sigma) \rightarrow (stm'_1, \sigma') \\
(stm_1; stm_2, \sigma) \rightarrow (stm'_1; stm_2, \sigma')
\]

**Nil**

\[
(nil; stm_2, \sigma) \rightarrow (stm_2, \sigma)
\]
Introduction
Semantic Framework
Co-Simulation Semantics

Formal Semantics
Plotkin’s SOS: While Rules

\[
\text{While true } \quad 
\begin{align*}
\llbracket B \rrbracket_\sigma &= \text{true} \\
(mk\text{-While}(B, stm), \sigma) &\rightarrow (stm; mk\text{-While}(B, stm), \sigma)
\end{align*}
\]

\[
\text{While false } \quad 
\begin{align*}
\llbracket B \rrbracket_\sigma &= \text{false} \\
(mk\text{-While}(B, stm), \sigma) &\rightarrow (\text{nil}, \sigma)
\end{align*}
\]
Outline

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

Type definition:

\[ Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event-\text{set} \times SysTag \]

- **DE**: Discrete-event simulator
- **CT**: Continuous-time simulator
- **\(\Sigma_o\)**: Shared variables between the simulators
- **Time**: The current time
- **Time**: The time bound on the CT execution
- **Event-\text{set}**: Events generated by CT
- **SysTag**: The simulator that took the last step
Co-Simulation
System Configuration

Type definition:

\[ \text{Config} = \text{DE} \times \text{CT} \times \Sigma_o \times \text{Time} \times \text{Time} \times \text{Event-set} \times \text{SysTag} \]

- **DE**: Discrete-event simulator
- **CT**: Continuous-time simulator
- **\( \Sigma_o \)**: Shared variables between the simulators
- **\( \text{Time} \)**: The current time
- **\( \text{Time} \)**: The time bound on the CT execution
- **\( \text{Event-set} \)**: Events generated by CT
- **\( \text{SysTag} \)**: The simulator that took the last step
Type definition:

\[ Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event-set \times SysTag \]

- \( DE \): Discrete-event simulator
- \( CT \): Continuous-time simulator
- \( \Sigma_o \): 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
Type definition:

\[ \text{Config} = DE \times CT \times \Sigma_o \times Time \times Time \times \text{Event-set} \times \text{SysTag} \]

- **DE**: Discrete-event simulator
- **CT**: Continuous-time simulator
- **\(\Sigma_o\)**: 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:

\( \text{Config} = \text{DE} \times \text{CT} \times \Sigma_o \times \text{Time} \times \text{Time} \times \text{Event-set} \times \text{SysTag} \)

- \( \text{DE} \): Discrete-event simulator
- \( \text{CT} \): Continuous-time simulator
- \( \Sigma_o \): Shared variables between the simulators

\[ \Sigma_o = \text{Id}_v \xrightarrow{m} (\text{SharedValue} \times \text{SysTag}) \]

- \( \text{Time} \): The current time
- \( \text{Time} \): The time bound on the CT execution
- \( \text{Event-set} \): Events generated by CT
- \( \text{SysTag} \): The simulator that took the last step
Type definition:

\[ \text{Config} = \text{DE} \times \text{CT} \times \Sigma_o \times \text{Time} \times \text{Time} \times \text{Event-set} \times \text{SysTag} \]

- \text{DE}: Discrete-event simulator
- \text{CT}: Continuous-time simulator
- \( \Sigma_o \): Shared variables between the simulators
- \text{Time}: The current time
- \text{Time}: The time bound on the CT execution
- \text{Event-set}: Events generated by CT
- \text{SysTag}: The simulator that took the last step
Type definition:

\[ Config = DE \times CT \times \Sigma_o \times Time \times Time \times Event-set \times SysTag \]

- **DE**: Discrete-event simulator
- **CT**: Continuous-time simulator
- **\(\Sigma_o\)**: 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
Type definition:

\[ \text{Config} = \text{DE} \times \text{CT} \times \Sigma_o \times \text{Time} \times \text{Time} \times \text{Event-set} \times \text{SysTag} \]

- \text{DE}: Discrete-event simulator
- \text{CT}: Continuous-time simulator
- \Sigma_o: Shared variables between the simulators
- \text{Time}: The current time
- \text{Time}: The time bound on the CT execution
- \text{Event-set}: Events generated by CT
- \text{SysTag}: The simulator that took the last step
Co-Simulation

System Configuration

Type definition:

\[ \text{Config} = \text{DE} \times \text{CT} \times \Sigma_0 \times \text{Time} \times \text{Time} \times \text{Event-set} \times \text{SysTag} \]

- \text{DE}: Discrete-event simulator
- \text{CT}: Continuous-time simulator
- \Sigma_0: Shared variables between the simulators
- \text{Time}: The current time
- \text{Time}: The time bound on the CT execution
- \text{Event-set}: Events generated by CT
- \text{SysTag}: The simulator that took the last step
The main co-simulation transition relations has the type:

\[ \xrightarrow{cs} : Config \times Config \]

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

\[ \langle DE \rangle \xrightarrow{cs} \langle CT \rangle \xrightarrow{cs} \langle DE \rangle \xrightarrow{cs} \langle CT \rangle \xrightarrow{cs} \ldots \]
Co-Simulation
DE Step

Co-Sim DE Step

\[(\textit{de}, \sigma_o, \tau, \textit{events}) \xrightarrow{\text{de}} (\textit{de}', \sigma_s, \tau_b')\]

\[\sigma'_o = \text{mergeStates}(\sigma_o, \sigma_s)\]

\[(\textit{de}, \textit{ct}, \sigma_o, \tau, \tau_b, \textit{events}, \langle\textit{CT}\rangle) \xrightarrow{\text{cs}} (\textit{de}', \textit{ct}, \sigma'_o, \tau, \tau'_b, \textit{events}, \langle\text{DE}\rangle)\]
Co-Simulation
CT Step

\[
\begin{align*}
(\textit{ct}, \sigma_o, \tau_b) \xrightarrow{\text{ct}} & (\textit{ct}', \sigma_s, \tau', \textit{events}') \\
\sigma'_o &= \text{mergeStates}(\sigma_o, \sigma_s) \\
(\textit{de}, \textit{ct}, \sigma_o, \tau, \tau_b, \textit{events}, \langle \text{DE} \rangle) \xrightarrow{\text{cs}} & (\textit{de}, \textit{ct}', \sigma'_o, \tau', \tau_b, \textit{events}', \langle \text{CT} \rangle)
\end{align*}
\]
CT Transition Relation

\[
\mathbf{ct} : (CT \times \Sigma_o \times Time) \times (CT \times \Sigma_o \times Time \times Event-set)
\]

CT Solver

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

\[ \text{de}: (DE \times \Sigma_o \times Time \times \text{Event-set}) \times (CT \times \Sigma_o \times Time) \]

DE Interpreter

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

**DE big step**

\[
\begin{align*}
    d e^1 &= updateDEFromShared(d e, \sigma_o) \\
    d e^2 &= commitPendingValuesAndUpdateTime(d e^1, \tau) \\
    d e^3 &= createPeriodicAndEventThreads(d e^2, events) \\
    d e^4 &= doContextSwitches(d e^3) \\
    d e^4 &\xrightarrow{deexec} d e^5 \\
    (\sigma_s, \tau_b) &= extractValuesAndMinDurationFromDE(d e^5) \\
    (d e, \sigma_o, \tau, events) &\xrightarrow{de} (d e^5, \sigma_s, \tau_b)
\end{align*}
\]
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