Semantic adaptation for FMI Co-simulation

Bart Meyers, Joachim Denil, Casper Thule, Kenneth Lausdahl
Peter Gorm Larsen, Hans Vangheluwe, Paul De Meulenaere


March, 2018
Example – Original System

Example – Co-simulation
FMU (Conceptual) Internals

Model
\[ \dot{x}_1 = F_1(x_1, u_1) \]
\[ y_1 = G_1(x_1, u_1) \]

Solver
\[ x(t + h) = x(t) + F(x(t), u(t)) \times h \]

Input Approximation
\[ u(t) = \phi_u(t, u(n \cdot H), u((n - 1) \cdot H), \ldots) \]

Orchestrator

March, 2018
Motivation for Semantic Adaptation

• Quick and sound way of adapting the behaviour of an interconnected set of FMUs
  • Data conversion
  • Interaction protocol modification
    • Time triggered vs Event triggered execution
  • Capability adaptation
• Support advanced co-simulation in importing tools
Example: Capability Interaction


March, 2018
Example: Capability Interaction

http://doi.org/10.1007/978-3-662-44926-4\_6
March, 2018
Example: Capability Interaction

Model | Solver
---|---
Input Approximation

Orchestrator


https://github.com/into-cps/case-study_mass-springer-damper

March, 2018
Example: Capability Conflict

Gauss-seidel orchestrator ↔ Interpolation

$t := t + H$

March, 2018
Power Window Case Study


Semantic Adaptation

- Actions by which the **behavior** of an original set of interconnected FMUs is **altered**, following the **transparency** and **modularity** principles.

**How?**

![Diagram of Semantic Adaptation]
Semantic Adaptation

• Actions by which the behavior of an original set of interconnected FMUs is altered, following the transparency and modularity principles.

How?

March, 2018
A DSL for Semantic Adaptation

```plaintext
semantic adaptation reactive moore ControllerSA controller_sa
at "./path/to/ControllerSA.fmu"

for inner func Controller ctrl
at "./path/to/LazySA.fmu"
with input ports obj_detected, passenger_up, passenger_down
with output ports up, down, stop

input ports armature_current -> ctrl.obj_detected,
    passenger_up -> ctrl.passenger_up,
    passenger_down -> ctrl.passenger_down,
    passenger_stop -> ctrl.passenger_stop,
    driver_up -> ctrl.driver_up,
    driver_down -> ctrl.driver_down,
    driver_stop -> ctrl.driver_stop

output ports u, d
```
A DSL for Semantic Adaptation

control rules {
    var step_size := H;
    var aux_obj_detected := false;
    var crossedTooFar := false;
    if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T) 
        and (not is_close(f_v, T, RTOL, ATOL) and f_v > T)) {
        crossedTooFar := true;
        var negative_value := p_v - T;
        var positive_value := f_v - T;
        step_size := (H * (-negative_value)) / (positive_value - negative_value);
    } else {
        if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T) 
            and is_close(f_v, T, RTOL, ATOL)) {
            c := true;
        }
    }
    if (not crossedTooFar) {
        step_size := do_step(ctrl, t, H);
    }
    if (is_close(step_size, H, RTOL, ATOL)) {
        p_v := f_v;
    }
    return step_size;
}

March, 2018
A DSL for Semantic Adaptation

in var  f_v := INIT_V;
in rules  
  true -> 
    f_v := controller_sa.armature_current;
  } -> 
    ctrl.obj_detected := c;
};

out rules 
  ctrl.up -> 
    } -> {controller_sa.u := 1.0; }
  not ctrl.up -> 
    } -> {controller_sa.u := 0.0; }
  ctrl.down -> 
    } -> {controller_sa.d := 1.0; }
  not ctrl.down -> 
    } -> {controller_sa.d := 0.0; }
  ctrl.stop -> 
    } -> {controller_sa.u := 0.0 ; controller_sa.d := 0.0; };

March, 2018
Summary & Future Work

- Motivation for semantic adaptations
- What are semantic adaptations
- How to implement them
- TODO: Higher level adaptations
Thank you!

Questions?