Status of Clocked Synchronous and State Machines Modeling in OpenModelica

Bernhard Thiele (Linköping University)
Lennart Ochel (FH Bielefeld)
Clocked Synchronous Modeling: Sampled Control

```
// time-discrete controller
when Clock() then
   E*xd = A*previous(xd) + B*yd;
   ud = C*previous(xd) + D*yd;
end when;
```

```
// hold
u = hold(ud)
```

```
// plant
0 = f(der(x), x, u);
y = g(x);
```

\[ yd(t_i) = y(t_i) \]
\[ E \cdot xd(t_i) = A \cdot xd(t_{i-1}) + B \cdot yd(t_i) \]
\[ ud(t_i) = C \cdot xd(t_{i-1}) + D \cdot yd(t_i) \]
\[ u(t) = ud(t_i), \quad t_i \leq t < t_{i+1} \]

Clocked variables \( r(t_i) \) are of base type Real, Integer, etc. They are uniquely associated with a clock \( c(t_i) \). Can only be accessed when its clock is active (ticks).

Clock variables \( c(t_i) \) are of base type Clock. They are defined by constructors such as \( \text{Clock}(3) \) or by clock operators relatively to other clocks.

Continuous variables are Real numbers defined as piecewise continuous functions of time.

Piecewise-constant variables \( m(t) \) are constant inside each \( t_i \leq t < t_{i+1} \).
Sampled control in Modelica 3.2

```modelica
event = sample(0,3);
when event then
    xd = A*pre(xd) + B*y;
    u = C*pre(xd) + D*y;
end when;
```

- Variables \( xd, u \) are piecewise-constant variables that change values at the sampling event.
- The ’old’ sample is *completely different* than the ’new’ clocked sample!
Modelica_Synchronous Library Example

Library for precise and convenient definition and synchronization of multi-rate data systems.
Modelica_Synchronous: Cascaded Control
# New Language Elements

| **Clock Constructors** | Clock(); Clock(intervalCounter, resolution);  
|                        | Clock(interval); Clock(condition, startInterval);  
|                        | Clock(c, solverMethod); |

| **Base-clock conversion operators** | sample(u,c); hold(u) |

| **Sub-clock conversion operators** | subSample(u,factor); superSample(u,factor);  
|                                   | shiftSample(u,shiftCounter,resolution);  
|                                   | backSample(u,backCounter,resolution); noClock(u) |

| **Other operators** | previous(u); interval(u) |
Status of implementation

• **Front-end:** Full support
  – All models flatten

• **Back-end:** Partial support
  – Prototypical implementation except for *Clocked Discretized Continuous-Time Partition, i.e.,* no “**Clock**(*c, solverMethod*)”
  – Simulation: 34 from 68 tests compile and execute

• **Run-time:** Poor support
  – Verified Simulation: 11 from 68 tests succeed
  – **TODO:** Integrate synchronous features into event system
State Machines – Simple Example

- Equations are active if corresponding *clock* ticks. Defaults to periodic clock with 1.0 s sampling period
- “i” is a shared variable, “j” is a local variable. Transitions are “*delayed*” and enter states by “*reset*”
model Simple_NoAnnotations "Simple state machine"

inner Integer i(start=0);

block State1
  outer output Integer i;
  output Integer j(start=10);
  equation
    i = previous(i) + 2;
    j = previous(j) - 1;
end State1;

State1 state1;

block State2
  outer output Integer i;
  equation
    i = previous(i) - 1;
end State2;

State2 state2;

equation
  transition(state1, state2, i > 10, immediate=false);
  transition(state2, state1, i < 1, immediate=false);
initialState(state1);
end Simple_NoAnnotations;
What happened since OM Workshop 2015?

• Introduced a dedicated symbolic representation for state machines extending the flat Modelica representation of OMC (see paper at Int. Modelica Conference 2015)

• Rewrote most of the transformation code utilizing the new representation

• Merged the work into the master development branch

→ Highly improved and fairly complete prototype
Status of Implementation

- Involved examples can be simulated
- Still using a workaround for compensating clocked synchronous issues: All SM related equation are simply wrapped in a when-equation with 1 s sampling period
- This restriction can be easily removed as soon as clocked synchronous constructs are working reasonably well

MLS 3.3, Section 17.3.7 works
Demonstration
Conclusion and Future Plans

• A lot of progress in the support of clocked synchronous and state machine features

• Clocked synchronous features plans:
  – Still need to fix many bugs
  – Better handling of discrete-time (result files/plots)
  – Support *Clocked Discretized Continuous-Time Partitions*

• State Machines plans:
  – Support Merging Connections to Multiple Outputs
  – Get rid of workaround when synchronous is ready
Bernhard Thiele (Linköping University)
Lennart Ochel (FH Bielefeld)

www.liu.se