Events

Events are ordered in time and form an event history

- A point in time that is instantaneous, i.e., has zero duration
- An event condition that switches from false to true in order for the event to take place
- A set of variables that are associated with the event, i.e., are referenced or explicitly changed by equations associated with the event
- Some behavior associated with the event, expressed as conditional equations that become active or are deactivated at the event. Instantaneous equations is a special case of conditional equations that are only active at events.
Hybrid Modeling

Hybrid modeling = continuous-time + discrete-time modeling

- A point in time that is instantaneous, i.e., has zero duration
- An event condition so that the event can take place
- A set of variables that are associated with the event
- Some behavior associated with the event, e.g. conditional equations that become active or are deactivated at the event

Event creation – if

if-equations, if-statements, and if-expressions

```model Diode "Ideal diode" extends TwoPin;
  Real s;
  Boolean off;
  equation
  off = s < 0;
  if off then
    v = s
  else
    v = 0;
  end if;
i = if off then 0 else s;
end Diode;
```

False if s<0
If-equation choosing equation for v
If-expression
Event creation – when

**when-equations**

```plaintext
when <conditions> then <equations> end when;
```

**Time event**

```plaintext
when time >= 10.0 then ...
end when;
```

**State event**

```plaintext
when sin(x) > 0.5 then ...
end when;
```

Equations only active at event times

Only dependent on time, can be scheduled in advance

Related to a state. Check for zero-crossing

Generating Repeated Events

The call `sample(t0,d)` returns true and triggers events at times `t0+i*d`, where `i=0,1,...`

```plaintext
model SamplingClock  
  Integer i;  
  discrete Real r;  
  equation  
    when sample(2,0.5) then  
      i = pre(i)+1;  
      r = pre(r)+0.3;  
    end when;  
  end SamplingClock;
```

Variables need to be discrete

Creates an event after 2 s, then each 0.5 s

`pre(...)` takes the previous value before the event.
Reinit – Discontinuous Changes

The value of a continuous-time state variable can be instantaneously changed by a reinit-equation within a when-equation.

```model BouncingBall "the bouncing ball model"  
  parameter Real g=9.81; //gravitational acc.  
  Real height(start=10),velocity(start=0);  
  equation  
    der(height) = velocity;  
    der(velocity)=-g;  
  when height<0 then  
    reinit(velocity, -c*velocity);  
  end when;  
end BouncingBall;```

Reinit "assigns" continuous-time variable velocity a new value.

initial and terminal events

Initialization actions are triggered by initial()

Actions at the end of a simulation are triggered by terminal()
Terminating a Simulation

There `terminate()` function is useful when a wanted result is achieved and it is no longer useful to continue the simulation. The example below illustrates the use:

```model terminationModel
  Real y;
equation
  y = time;
  when y > 5 then
    terminate("The time has elapsed 5s");
  end when;
end terminationModel;
```

```
simulate(terminationModel, startTime = 0, stopTime = 10)
```

Simulation ends before reaching time 10

---

Expressing Event Behavior in Modelica

`if-equations, if-statements, and if-expressions` express different behavior in different operating regions

```model Diode "Ideal diode"
  extends TwoPin;
  Real s;
  Boolean off;
equation
  off = s < 0;
  if off then
    v = s;
  else
    v = 0;
  end if;
i = if off then 0 else s;
end Diode;
```

```when-equations` become active at events

```when <condition> then
  <equations>
end when;
```

```equation
  when x > y.start then
    ...
```
Event Priority

Erroneous multiple definitions, single assignment rule violated

```model WhenConflictX  // Erroneous model: two equations define x
discrete Real x;
equation
when time > 2 then
  x = pre(x)+1.5;
end when;
when time > 1 then
  x = pre(x)+1;
end when;
end WhenConflictX;
```

Using event priority to avoid erroneous multiple definitions

```model WhenPriorityX
discrete Real x;
equation
when time >= 2 then // Higher priority
  x = pre(x)+1.5;
elsewhen time >= 1 then // Lower priority
  x = pre(x)+1;
end when;
end WhenPriorityX;
```

Obtaining Predecessor Values of a Variable Using \texttt{pre()}\texttt{}

At an event, \texttt{pre(y)} gives the previous value of \texttt{y} immediately before the event, except for event iteration of multiple events at the same point in time when the value is from the previous iteration

- The variable \texttt{y} has one of the basic types \texttt{Boolean}, \texttt{Integer}, \texttt{Real}, \texttt{String}, or \texttt{enumeration}, a subtype of those, or an array type of one of those basic types or subtypes
- The variable \texttt{y} is a discrete-time variable
- The \texttt{pre} operator can \textit{not} be used within a function
Detecting Changes of Boolean Variables Using `edge()` and `change()`

Detecting changes of boolean variables using `edge()`

The expression `edge(b)` is true at events when `b` switches from false to true.

Detecting changes of discrete-time variables using `change()`

The expression `change(v)` is true at instants when `v` changes value.

Creating Time-Delayed Expressions

Creating time-delayed expressions using `delay()`

In the expression `delay(v, d)` `v` is delayed by a delay time `d`.
A Sampler Model

```model Sampler
  parameter Real sample_interval = 0.1;
  Real x(start=5);
  Real y;
  equation
d  er(x) = -x;
  when y = x;
end when;
end Sampler;
```

simulate(Sampler, startTime = 0, stopTime = 10)
plot({x,y})

Discontinuous Changes to Variables at Events via When-Equations/Statements

The value of a **discrete-time** variable can be changed by placing the variable on the left-hand side in an equation within a **when-equation**, or on the left-hand side of an assignment statement in a **when-statement**

The value of a **continuous-time** state variable can be instantaneously changed by a **reinit-equation** within a **when-equation**

```model BouncingBall "the bouncing ball model"
  parameter Real g=9.18; //gravitational acc.
  parameter Real c=0.90; //elasticity constant
  Real x(start=0), y(start=10);
  equation
der(x) = y;
der(y) = -g;
  when x<0 then
    reinit(y, -c*y);
  end when;
end BouncingBall;
```
A Mode Switching Model Example

Elastic transmission with slack

DC motor transmission with elastic backlash

A finite state automaton

SimpleElastoBacklash model

A Mode Switching Model Example cont’

partial model SimpleElastoBacklash
  Boolean backward, slack, forward;  // Mode variables
  parameter Real  c = 1.e5     "Size of backlash region";
  parameter Real  phi_rel0 = 0 "Angle when spring exerts no torque";
  Real         phi_rel      "Relative rotation angle betw. flanges";
  Real         phi_dev      "Angle deviation from zero-torque pos";
  Real         tau          "Torque between flanges";
  equation
    phi_rel   = flange_b.phi - flange_a.phi;
    phi_dev   = phi_rel - phi_rel0;
    backward  = phi_rel < -b/2;    // Backward angle gives torque tau<0
    forward   = phi_rel > b/2;     // Forward angle gives torque tau>0
    slack     = not (backward or forward); // Slack angle gives no torque
    tau = if forward then
          c*(phi_dev - b/2)  // Forward angle gives
            else // Backward angle gives
               c*(phi_dev + b/2) // positive driving torque
               else // negative braking torque
                  0; // Slack gives
            end // zero torque
A Mode Switching Model Example cont’

Relative rotational speed between the flanges of the Elastobacklash transmission

We define a model with less mass in inertia2 (J=1), no damping d=0, and weaker string constant c=1e-5, to show even more dramatic backlash phenomena.

The figure depicts the rotational speeds for the two flanges of the transmission with elastic backlash.

Water Tank System with PI Controller

```model TankPI
  LiquidSource source(flowLevel=0.02);
  Tank tank(area=1);
  PIcontinuousController piContinuous(ref=0.25);
  equation
    connect(source.qOut, tank.qIn);
    connect(source.tOut, piContinuous.cOut);
    connect(tank.tActuator, piContinuous.cIn);
  end TankPI;
```

```model Tank
  ReadSignal tOut; // Connector, reading tank level
  ActSignal tInp; // Connector, actuator controlling input flow
  parameter Real flowVout = 0.01; // [m3/s]
  parameter Real area = 0.5; // [m2]
  Real h(start=0); // tank level [m]
  Real qIn; // flow through input valve [m3/s]
  Real qOut; // flow through output valve [m3/s]
  equation
    der(h) = (qIn-qOut)/area; // mass balance equation
    qOut = if time>100 then flowVout else 0;
    qIn = flowGain*tInp.act;
    tOut.val = h;
  end Tank;
```
Water Tank System with PI Controller – cont'

partial model BaseController
parameter Real Ts(unit = "s") = 0.1 "Time period between discrete samples";
parameter Real K = 2 "Gain";
parameter Real T(unit = "s") = 10 "Time constant";
ReadSignal cIn
ActSignal cOut
parameter Real ref "Reference level";
Real error "Deviation from reference level";
Real outCtr "Output control signal";
equation
    error = ref - cIn.val;
    cOut.act = outCtr;
end BaseController;

model PIdiscreteController extends BaseController(K = 2, T = 10);

equation
    when sample(0, Ts) then
    x = x + error * Ts / T;
    outCtr = K * (x + error);
end when;
end PIdiscreteController;

model PIDcontinuousController extends BaseController(K = 2, T = 10);

equation
    der(x) = error/T;
    y = T * der(error);
    outCtr = K * (error + x + y);
end PIDcontinuousController;

Concurrency and Resource Sharing

Dining Philosophers Example

model DiningTable
parameter Integer n = 5  "Number of philosophers and forks";
parameter Real sigma = 5  " Standard deviation for the random function";
// Give each philosopher a different random start seed
// Comment out the initializer to make them all hungry simultaneously.
Philosopher phil[n](startSeed=[1:n,1:n,1:n], sigma=fill(sigma,n));
Mutex mutex(n=0);
Fork fork[n];
equation
for i in 1:n loop
    connect(phil[i].mutexPort, mutex.port[i]);
    connect(phil[i].right, fork[i].left);
    connect(fork[i].right, phil[mod(i, n) + 1].left);
end for;
end DiningTable;