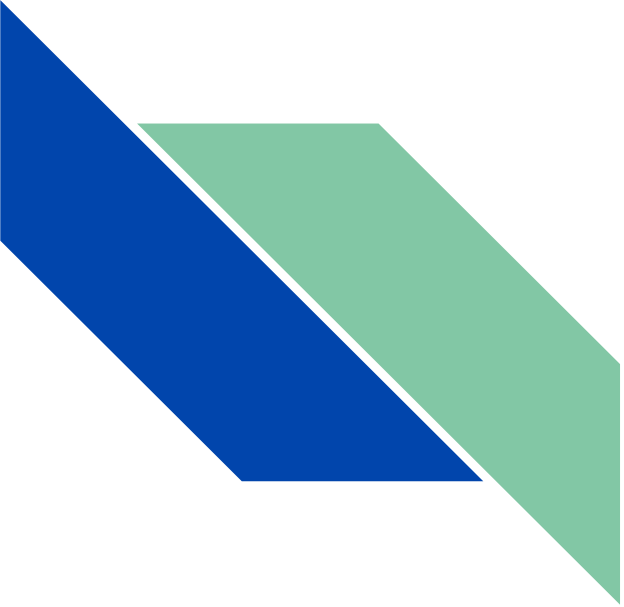


Discontinuous & Hybrid System Modeling

ECSE 4961/6961
Prof. Luigi Vanfretti

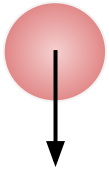
Overview

- What is a hybrid system?
- What is an event?
- Chattering
- Avoiding events
- Variable structures
  - Parameterized curves
  - State machines

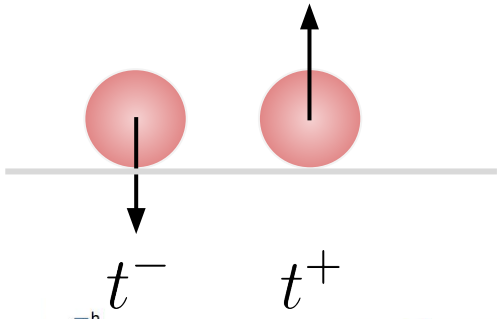
What is a Hybrid System?

- A hybrid system contains both continuous and discrete parts.
- For a bouncing ball we have:

A continuous part:
Ball in gravity field



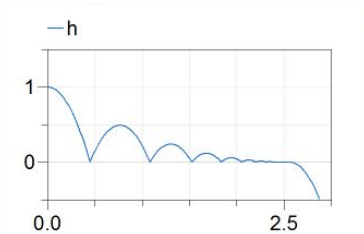
$$\frac{d^2x}{dt^2} = g$$



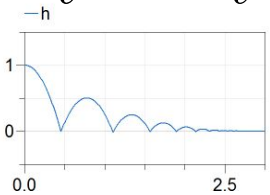
A discrete part:
Ball bounces back when hitting the surface

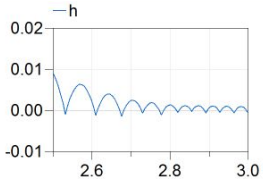
$$\frac{dx}{dt}(t^+) = -\frac{dx}{dt}(t^-)$$

DASSL

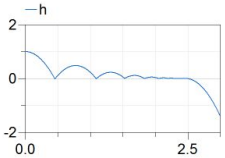


Euler
(default settings,
Tol = 0.0001,
Default Δt)





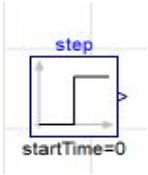
RKFix2
(default settings,
Tol = 0.0001,
Default Δt)



- Which one is the correct answer?
- What do we do about this? For an efficient and reliable handling of discontinuous dynamics, standard integrators need help!

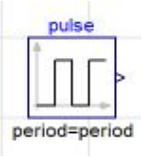
Common Discontinuities

```
y = offset +
(if time < startTime
then 0 else height);
```



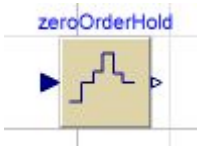
- Discontinuous input functions, e.g. a step, pulse, … in `Modelica.Blocks.Sources`

```
when integer((time - startTime)/period) >
pre(count) then
    count = pre(count) + 1;
    T_start = time;
end when;
```



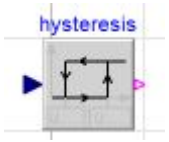
Notice → `if, then, else; when, integer(), and pre()`

- Sampled systems, e.g. for digital controllers, `Modelica.Discrete.ZeroOrderHold`




```
when {sampleTrigger, initial()} then
  ySample = u;
end when;
y = pre(ySample);
```

- Hysteresis, e.g. `Modelica.Blocks.Logical.Hysteresis`




```
assert(uHigh > uLow,"Hysteresis limits wrong (uHigh <= uLow)");
y = not pre(y) and u > uHigh or pre(y) and u >= uLow;
```

`integer()`, `pre()`, and `initial`, are **Event Operators.**

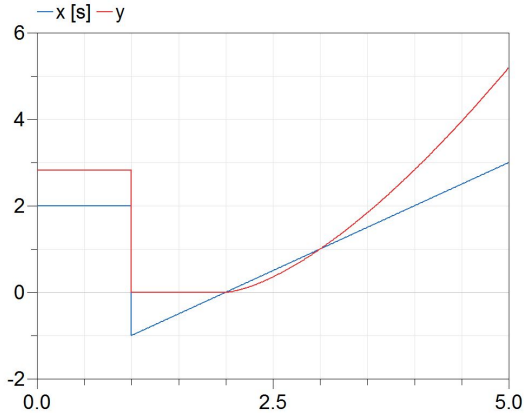
`assert()` is a **Special Operator**

The `smooth` operator

Usage	Description
`smooth(p, expr)`	When evaluated, the `smooth` operator simply returns the value of the expression `expr`. But it also acts as a guarantee that `expr` is `p` times continuously differentiable. For further details, see §3.7.3.2.

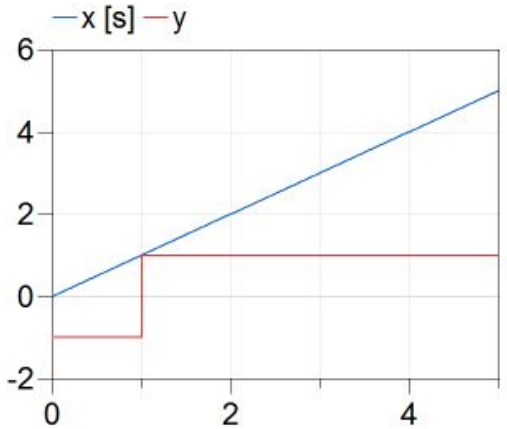
- Such discontinuities are very common.
- In the cases above:
  - The output jumps from one value to another, depending on input and possibly internal states.
- Note that the output from these blocks is not continuous at switching points:
  - What is the consequence? → cannot be differentiated
  - The `noEvent` and `smooth` operator may need to be used.
  - The homotopy operator may also be used when an "approximate" continuous form of the expression can be provided.
- More about this later in class.

```
model smooth_example_first_order
 Real x,y;
 parameter Real p=1;
equation
 x = if time<1 then 2 else time-2;
 y = smooth(1, noEvent(if x<0 then 0 else
sqrt(x)*x));
end smooth_example_first_order;
```



What is an Event?

```
model simple_event
  Real x,y;
equation
  x = time;
  y = if x>1 then 1 else -1;
end simple_event;
```



- An event corresponds to a discrete change in a system of equations.

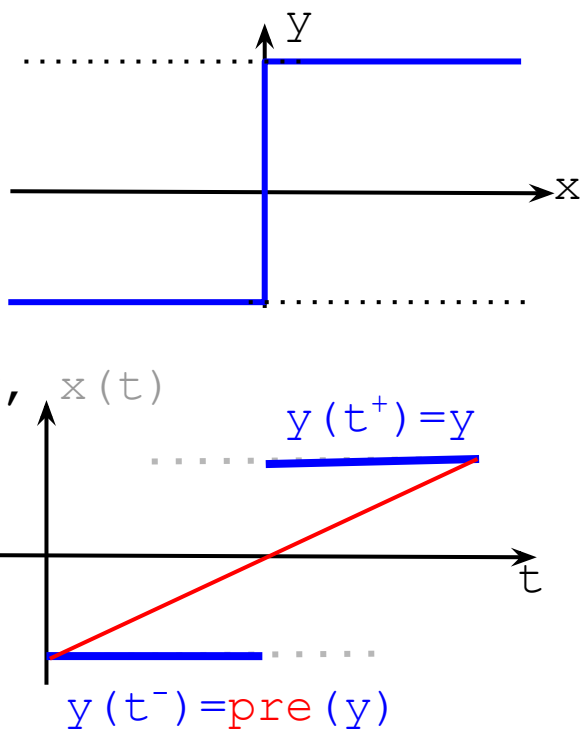
- The event is associated with”
  - A time point, when the change occurs
  - A set of equations
  - A condition, which activates one of the conditional equations

```
y = if x>1 then 1 else -1;
```

- Relations and Boolean expressions trigger events.
- Real variables that are changed during events are considered discrete.
- Integers and Booleans are always discrete.

Event Handling

```
y = if x>0 then 1 else -1;
```



During simulation:

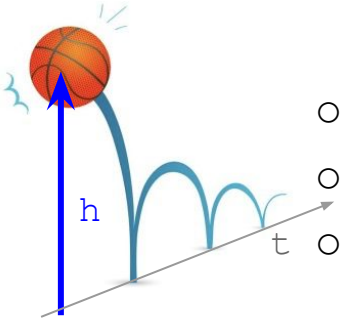
1. An event is generated when `x` crosses `0`.
2. The continuous integration is stopped
3. The time t_e for which $x(t_e)=0$ is searched using a crossing function.
   a. This requires that `y()` can also be evaluated beyond the event limit.
4. An initialization problem is solved at $t = t_e$
5. The continuous integration continues.

Time Events and State Events

- A **time event** is an event that **depends on the simulated time only**, an example is a step signal:

```
y = if time < t_start then y_0 else y_0 + y_step);
```

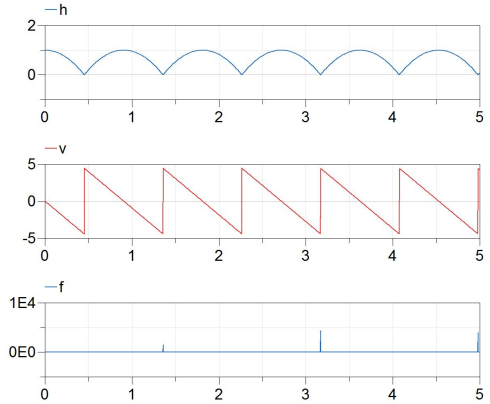
- The time for a time event is easy to find.
- The **state event** depends on one or more states of the model, for example the contact force in the bouncing ball:
  - The force needs to be calculated when the ball reaches the surface at *h=0*
  - A fixed time step integrator might miss this spot
  - This means that a variable time step integrator is needed
  - This also means that that this “state event” will require more time to be accurately accounted for by the solver.



State Event Example

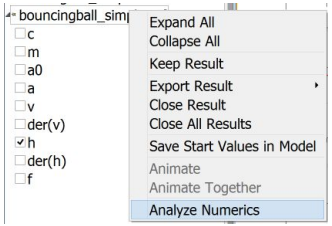
- The simplest model for the bouncing ball can be defined as
  - The motion of the ball is defined by the *height* above the ground and the vertical *velocity*.
- The ball moves continuously between bounces.
- An ideal ball would have a contact force equal to the gravitational force.
- What is the solver doing?
- What happens when “c” is changed?

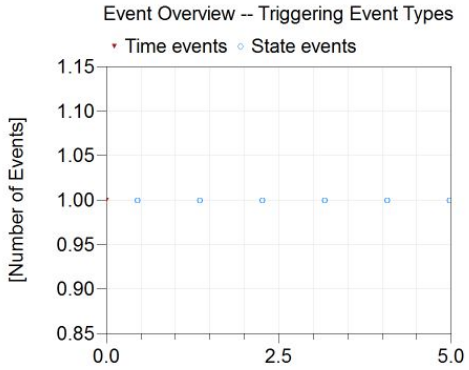
```
model bouncingball_simplest
  parameter Real c = 1e6;
  parameter Real m = 1;
    // mass
  parameter Real a0 = 9.81;
    // acceleration
  Real a(start=a0), v(start=0), h(start=1),
f;
equation
  der(h) = v;
  der(v) = a;
  f = m*(a+a0); // gravitational force
  f = if h > 0 then 0 else -c*h;
// the velocity is reversed, i.e. -c*h

end bouncingball_simplest;
```

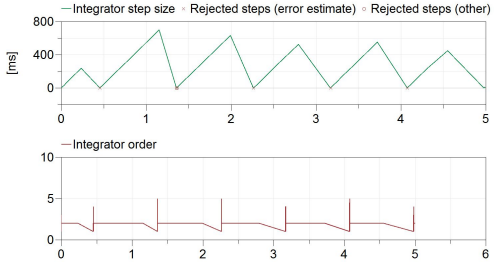


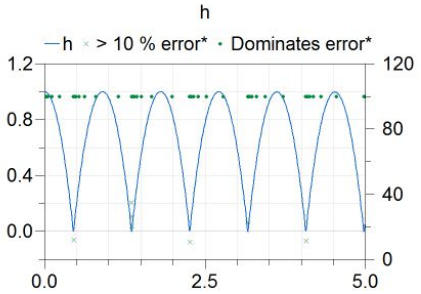
State Event Example

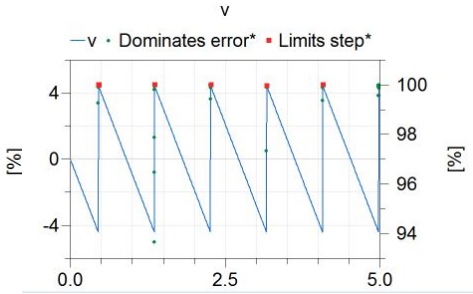
- On Setup>Debug> select:
  - Events during simulation
  - Events during initialization
  - Which states dominate error
- In Dymola use the option:
  - “Analyze numerics”
- This can help understanding the issues the solver faces.
  - Although one would think “h” is the issue, the variable that dominates the error is the velocity.





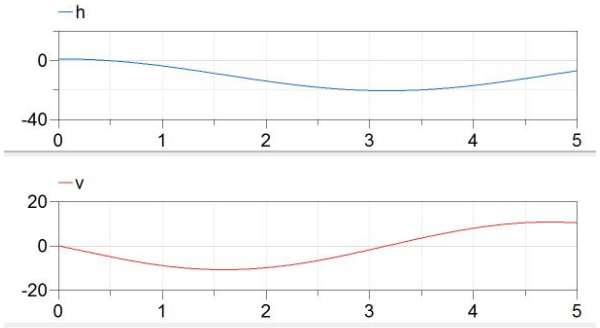


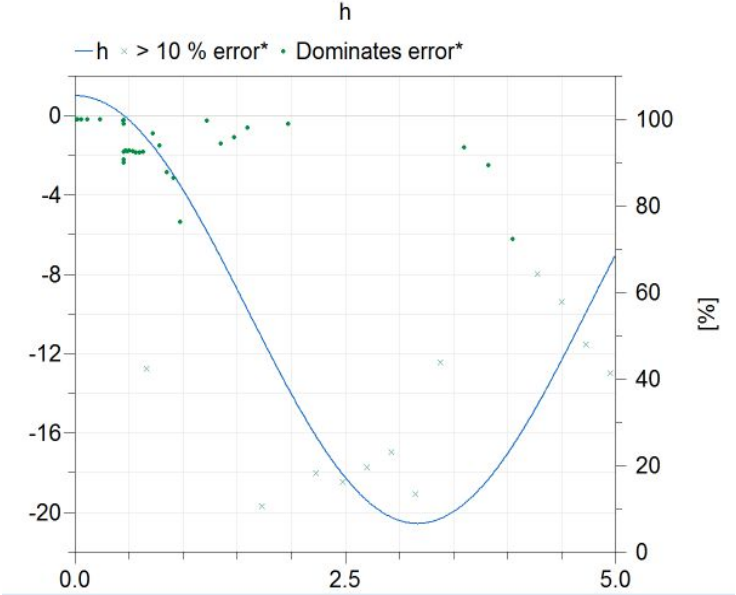


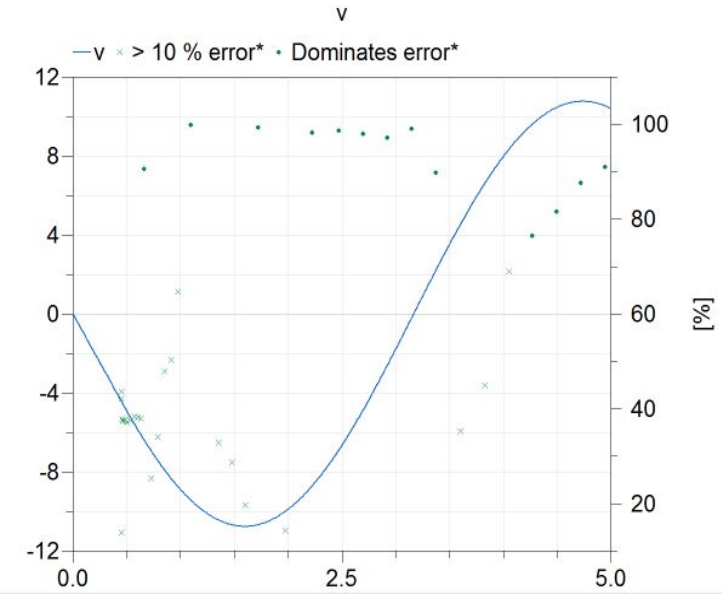


State Event Example

- What happens when “c” is changed?
  - Setting c=1
  - Is the state event detected?







Why Events?

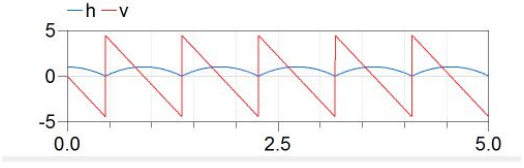
- **Detecting** and **triggering** an event ensures that the discontinuities are treated in a numerically sound way.

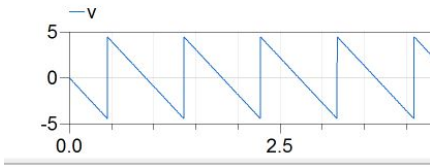
- If no event is generated at a time of discontinuity, **a small step size of the integrator is required** for a high accuracy.

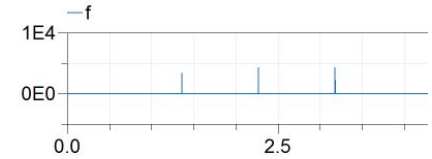
- In the case of the bouncing ball, missing an event would postpone the bouncing instant after the ball has reached the ground!

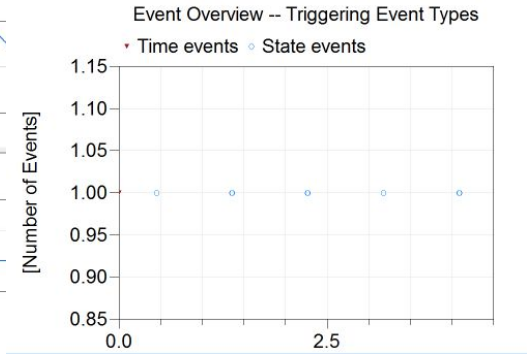
Why Events?

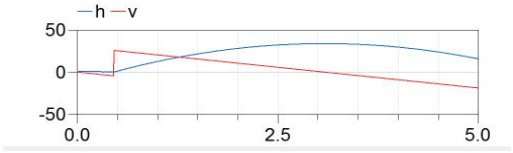
- In the case of the bouncing ball, missing an event would postpone the bouncing instant after the ball has reached the ground!
- Use Euler with step of 1e-6 and 0.001, and tolerance 0.0001

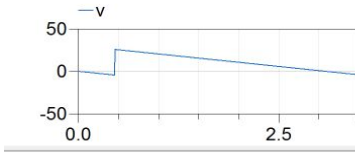


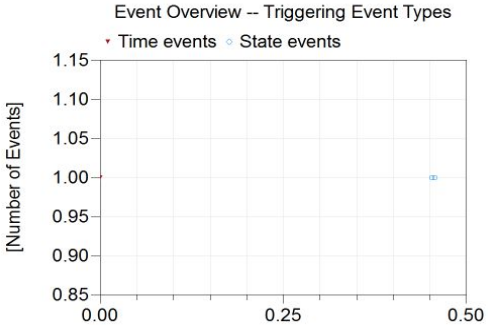












When-Statement

- Additional equations can be added during events using the **when** statement.
- These equations are **only active** during the event when the conditions become true, and are **deactivated** during continuous integrations.

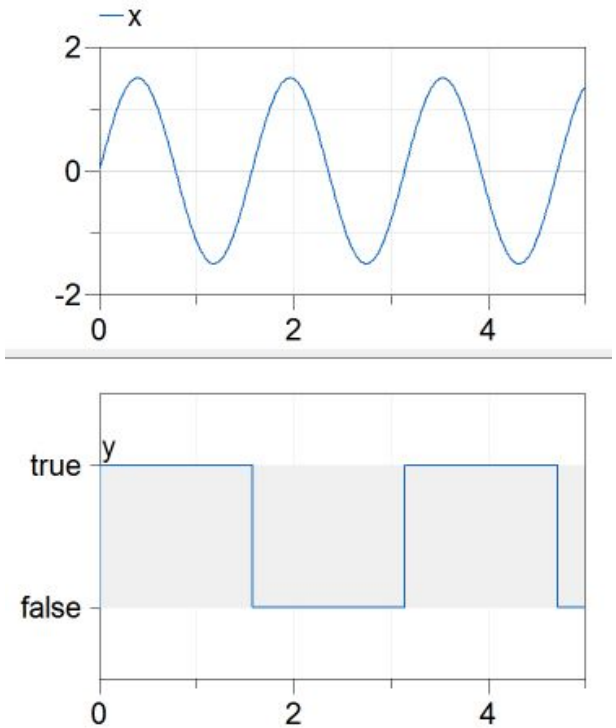
```
when expression then
     { statement ";" }
{ elsewhen expression then
     { statement ";" } }
end when;
```

```
when condition then
elsewhen condition then
end when;
```

- Restrictions
  - A when-statement shall not be used within a function
  - When-statements cannot be nested
  - When-statements may not occur inside while, if and for-clauses in algorithms.

Using Events - Example

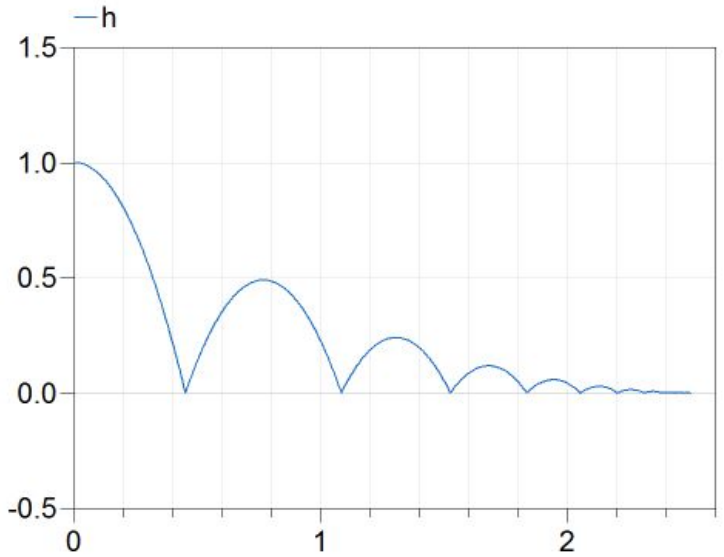
```
model WhenDemo
  parameter Real A=1.5, w=4;
  Real x(start=0);
  Boolean y;
equation
  x = A * sin(w*time);
  when x > 0 then
    y = not pre(y);
  end when
end WhenDemo;
```



Reinitialization

- State variables can be reinitialized during events using `reinit()`, e.g:

```
model BouncingBall_simple
  /* After around ~3 seconds h becomdes
negative due to numerical errors.

  */
  parameter Real e=0.7 "bounce coeff";
  parameter Real g=9.81 "gravity acc.";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";

equation
  der(h) = v;
  der(v) = -g;
  when h < 0 then
    reinit(v, -e*v);
  end when;
end BouncingBall_simple;
```




```
reinit(v, -e*v);     → change the direction
end when;            → when hitting the surface
```

What about Multiple Events?

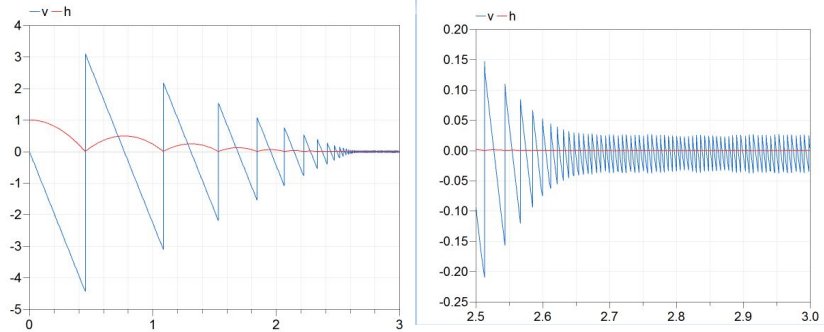
- In the case of multiple and simultaneous events, the synchronous data-flow principle restricts the use of the when statement:

```
equation //illegal example
when condition1 then
 close = true;
end when;
when condition2 then
 close = false;
end when;

equation //legal example
when condition1 then
 close = true;
elsewhen condition2 then
 close = false;
end when;
```

Synchronous Data Flow

- For the sake of **solvable** simulation models and **deterministic** results, Modelica is based on the synchronous data flow principle, that consists of:
- **Single assignment rule:** the total number of equations is at all times identical to the total number of unknown variables
- **Concurrency principle:** At every time instant, during continuous integration and at event instants, the active equations express relations between variables which have to be filled concurrently
  - (equations are not active if the corresponding if-branch, when-clause or block in which the equation is present is not active)
- **Synchronous principle:** events take no simulation time.
- All variables keep their actual values until these values are explicitly changed.
  - Variable values can be accessed at any time instant during continuous integration and at event instants.

A more advanced bouncing ball example



See: “A Modelica-based Format for Flexible Modelica Code Generation and Causal Model Transformations”

Link: here

- "If the ball is on the ground and has a negative velocity but the impact variable has not changed the velocity will be reset to zero and the boolean variable flying is set to false. The derivative of the velocity in the equation section is then also set to zero due to the if-expression"

```
model BouncingBall_advanced
  parameter Real e=0.7 "bounce coeff";
  parameter Real g=9.81 "gravity acc.";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";
  Boolean flying(start=true);
  Boolean impact;
  Real v_new;
equation
  impact = h <= 0.0;
  der(v) = if flying then -g else 0;
  der(h) = v;
  when {impact and v <= 0.0} then
    // edge() is expanded into (b and not pre(b))
    v_new = if edge(impact) then -e*pre(v) else 0;
    flying = v_new > 0;
    reinit(v, v_new);
  end when;
end BouncingBall_advanced;
```

Overview

- What is a hybrid system?
- What is an event?
- Chattering
- Avoiding events
- Variable structures
  - Parameterized curves
  - State machines

Chattering

Chattering

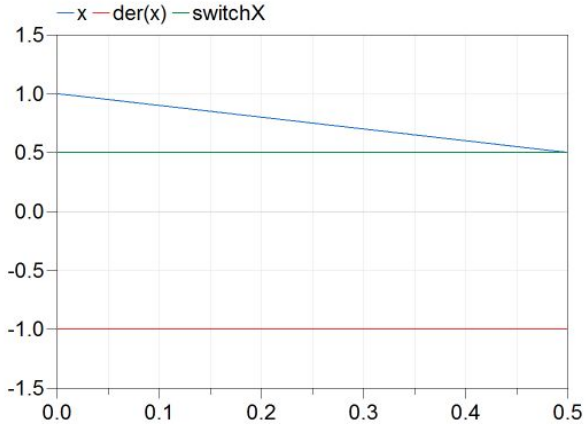
- There are many problems of physical significance that have no “clean” mathematical solution → care must be taken when posing the mathematical model.
  - **Fact:** a solution may fail to exist, especially when derivatives do not depend continuously on the state.
- Chattering is an undesirable phenomenon caused by rapid switching between different dynamics at an event barrier.
  - Very difficult to handle automatically
  - Potentially more than one discrete state involved
  - Switching may occur only for rare parameter values and operating points of model.
- Common problem in Hybrid models

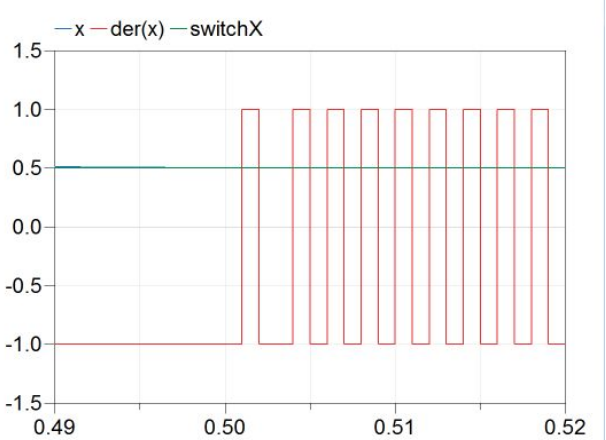
Chattering

- Simple chattering example:

```
model Chattering
    Real x(start = 1.0);
    parameter Real switchX = 0.5;
equation
  der(x) = if x> switchX then -1 else 1;
end Chattering;
```

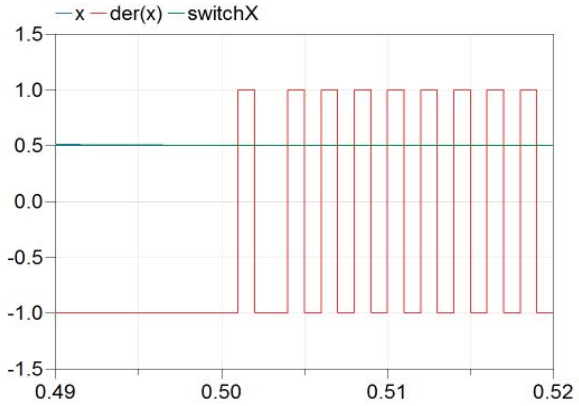
- Note that the derivative is not continuous across the event expression

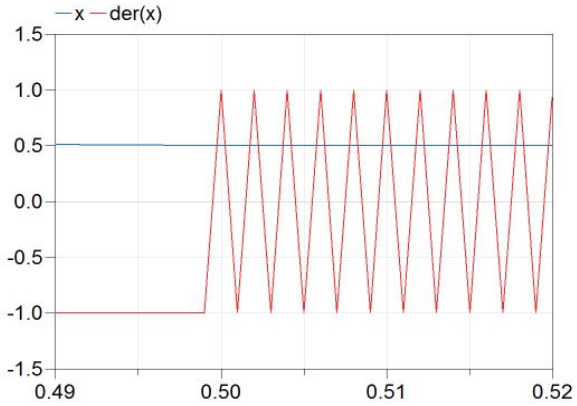


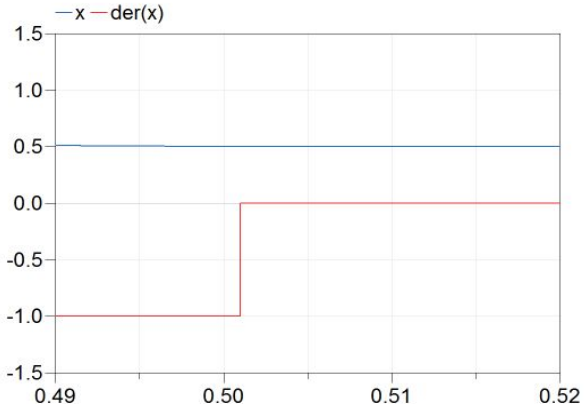


Chattering

- The solution obtained depends on:
  - (1) how the model is formulated
  - (2) steps taken in the model formulation to deal with the discontinuity

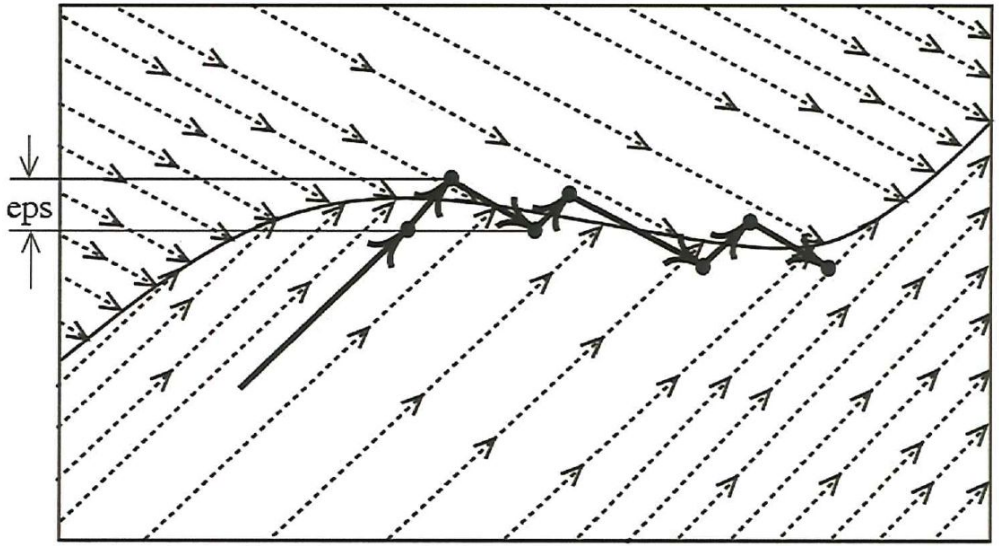






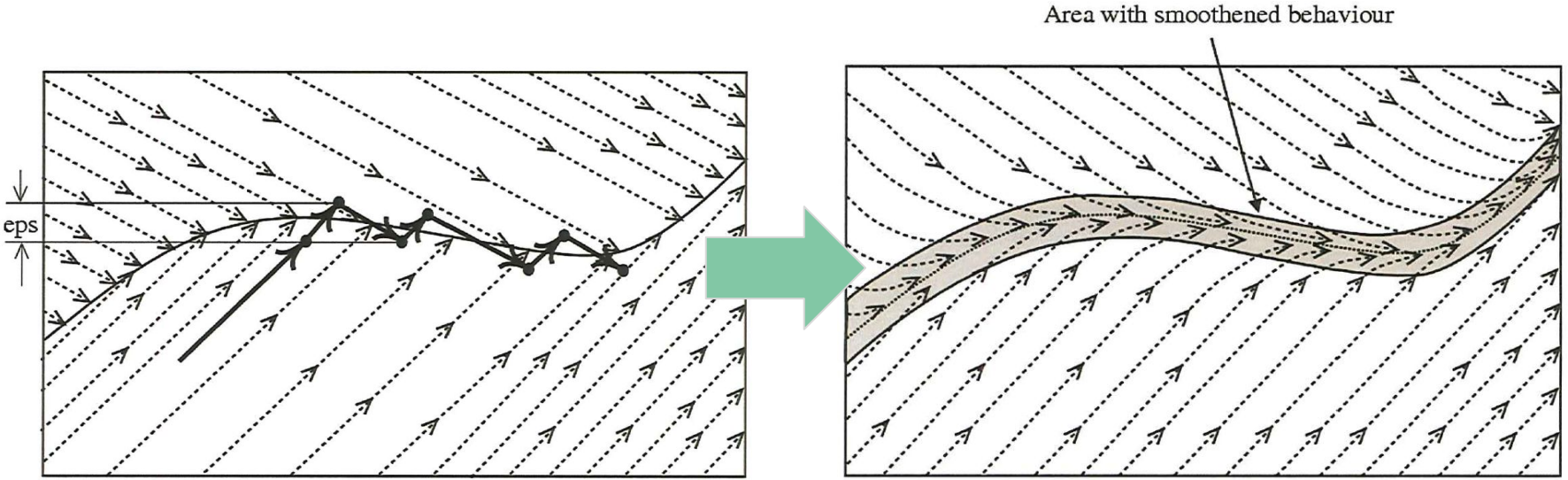
Chattering - Problem Origin

- The time integrator is chattering when it comes close to the event barrier
- Currently no known algorithm/solver for detecting chattering and automatically switching to Filippov-solution, except in trivial cases



Chattering - Workarounds

- Smoothen the discontinuity (choose different approximation for model)
- Find conditions for chattering explicitly and provide Filippov-solution and additional mode changes “by hand”.
  - Several ways to do this in Modelica.



Avoiding Events

Avoiding Events

- There are cases when events are not wanted, for example when preventing division by zero.
- Then built-in operators can be used, e.g.:

```
y1 = smooth(1, if x>eps then 1/x else 1/eps);
y2 = noEvent(if x>eps then 1/x else 1/eps);
y3 = 1/(max(x,eps));
```

- In some situations, event triggering cannot be avoided, Boolean and discrete variables always generate events → to avoid the event use:

```
Boolean x_is_small = noEvent(x<eps);
```

Avoiding Events

- noEvent()
  - Expressions are taken literally instead of generating crossing functions
  - Since there is no crossing function, there is no requirement that the expression can be evaluated beyond the event limit.
- The smooth() operator allows to define how many times an expression is continuously differentiable.

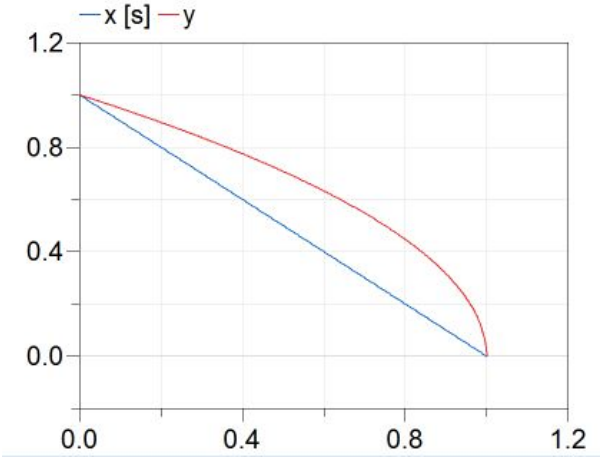
```
y = smooth(2, if x>0 then x^3 else x^4);
```

- min(), max(), abs() return continuous variables if the input is continuous, so no events are generated for these operators.

Using Noevent()

- Consider the following model:

```
model sqrtnew
  Real x;
  Real y;
equation
  x = 1-time;
  y = if x>0 then sqrt(x)
else 0;
end sqrtnew;
```



- This model will fail in Dymola, the reason is that:
  - sqrt(x) cannot be evaluated for **x<0**
  - The expression for y will generate an event
  - A crossing function is required
    - Must be able to evaluate sqrt(x) for any x around x = 0

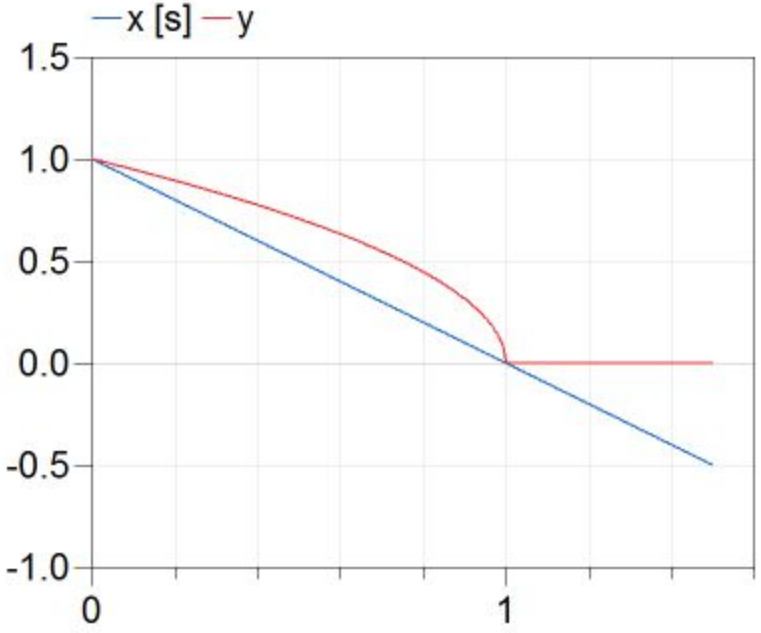
Using Noevent()

- Dymola can generate a crossing function.
- This is done automatically when applying noEvent():

```
y = noEvent(if x>0 then sqrt(x) else 0);
```

- The new model can be simulated:

```
model sqrtnewnoevent
  Real x;
  Real y;
equation
     x = 1-time;
     y = noEvent(if x>0 then sqrt(x) else
end sqrtnewnoevent;
```

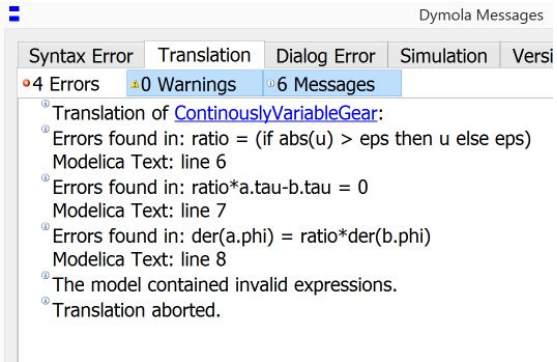


Being smooth()

- Consider the model:

```
model ContinouslyVariableGear
Modelica.Mechanics.Rotational.Interfaces.Flange_a a,b;
  input Real u;
  parameter Real eps = 1e-6;
equation
  ratio = if abs(u)>eps then u else eps;
  ratio * a.tau - b.tau = 0;
  der(a.phi) = ratio * der(b.phi);
end ContinouslyVariableGear;
```

- This model will fails Dymola because:
  - Ratio cannot be differentiated
  - Dymola cannot perform index reduction
  - Model will fail to translate
- It is then necessary to tell Dymola to ignore the discontinuity at u=0
  - Applying smooth() to the expression for ratio allows the model designer to tell Dymola to ignore the discontinuity at u=0.
  - Use noEvent() inside smooth().

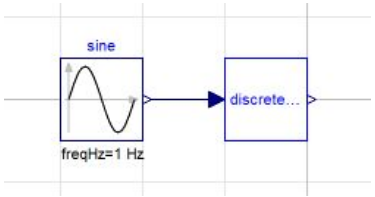


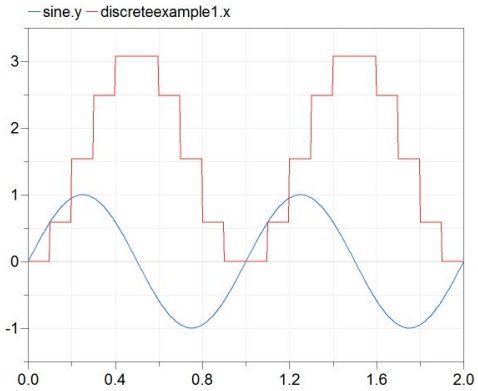
Sample Operator

- `sample ( start, interval)`
  - Returns true and triggers time events at time instants `start+i*interval (i=0,1,2,....)`.
  - During continuous integration the operator returns false always.
  - The starting time start and the sample interval interval need to be parameter expressions and need to have subtype of Real or Integer

```
model discreteexample
  Real x;
  parameter Real a=1, b=1;
  Modelica.Blocks.Interfaces.RealInput  u;
  Modelica.Blocks.Interfaces.RealOutput
y;
equation
  when sample(0,0.1) then
    x = a*pre(x)+b*pre(u);
  end when;
  y = x;
end discreteexample;
```

Variable x becomes discrete because of equation inside sample





Event Causing Operations

- Operators
  - initial()
  - terminal()
  - when (boolean)
  - sample(start, interval)
  - pre (Real)
  - edge (boolean)
  - change (Real)
  - reinit (Real, expr)
- See Modelica Reference library for operator definitions

- Mathematical Functions:
  - div(x,y)
  - mod(x,y)
  - rem(x,y)
  - ceil(x)
  - floor(x)
  - integer(x)

- smooth(integer,expr) (may generate event)

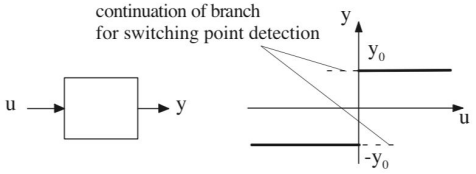
Operations Not Causing Events

- minReal,Real), max(Real,Real)
- Integer(Enumeration)
- abs(Real)
- sqrt(Real)
- sign(Real)
- noEvent(expr)

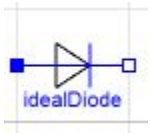
- smooth(integer,expr) (may generate event)

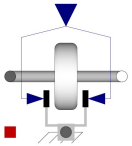
Discontinuous Components
Variable Structures

Discontinuous Components

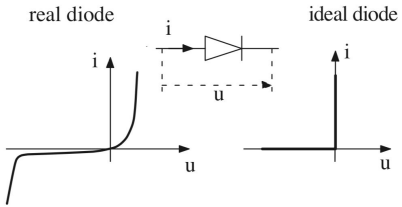


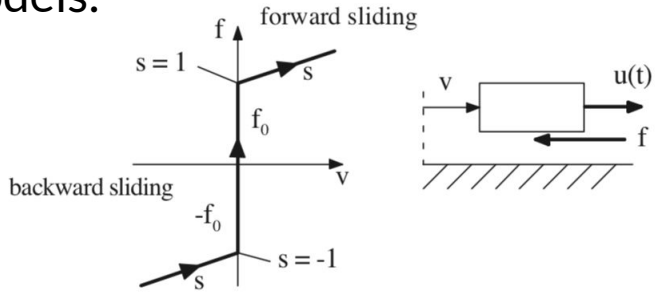
The equations and the equation structure of some models change depending on working conditions:





- Ideal diode, either voltage or current is constant
  - Modelica.Electrical.Ideal.IdealDiode
- Coulomb friction, either force or torque is constant, or speed is zero
  - Modelica.Mechanics.Rotational.Brake
- Two main ways to describe the singular models:
  - Parameterized curves
  - State machines

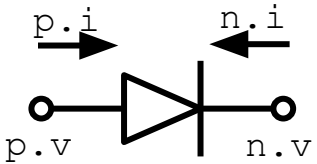


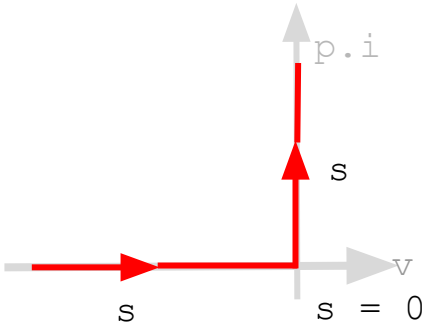


Otter M., Elmqvist H., and Mattsson S.E. (1999): Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle. CACSD'99, Aug. 22.-26, Hawaii.

Parameterized Curve - Declarative Description: Diode

- Define a variable `s`, and let `v` and i depend on `s`.
- Two equations then define `s` and the relation between `v` and `I`.

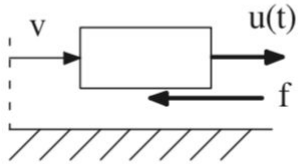




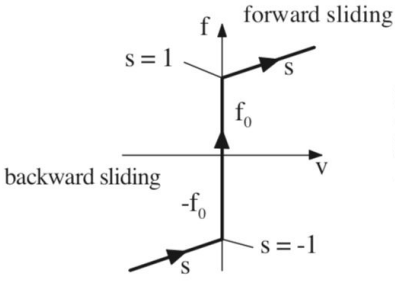

```
model Diode "Ideal diode"
  extends TwoPin;
  Real s;
  Boolean off;
equation
  off = s < 0;
  v = if off then s else 0;
  p.i = if off then 0 else s;
end Diode;
```

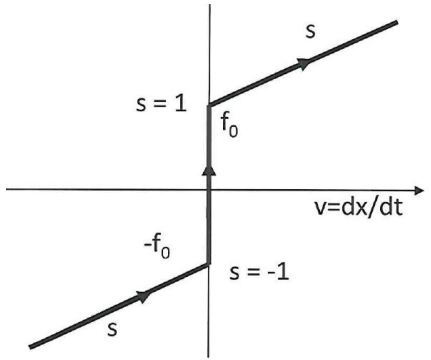
This is in a declarative description form.

Parameterized Curve: Friction



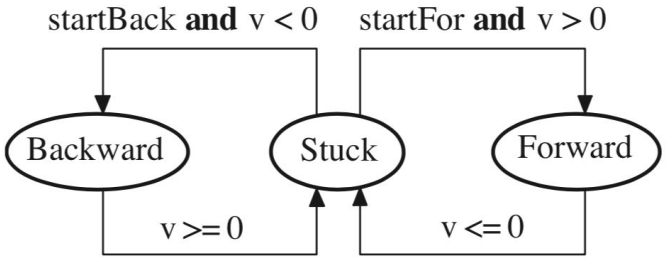
- Apply the corresponding method to a friction element, with v as the speed and f as the force.
- In this case, if friction is stuck, i.e. neither forward nor backward moving.
- The relative speed between the flanges v=0, and the relative position x is constant.
- This means Dymola has to handle behaviour like a
  - Gear - when stuck
  - Damper - when sliding
- But gear require index reduction (see lecture 7), while damper is just a force that depends on velocity.
- ***BUT:*** There is no known general method to handle the conditional index reduction!





Friction in Modelica Standard Library

- Write the parameterized curve on the force-acceleration level instead
- No differentiation is needed
- Information of what state the friction is in must be supplied
- Introduce a state machine to monitor the friction state



- If the mode is Backward or Forward, calculate the force depending on v (curve parameter is force)
- Else, compute the force so that acceleration is 0 (curve parameter is acceleration)

Note that this requires that the velocity and position can be calculated outside the friction model, at each side of it. It is thus not possible to connect two friction models directly to each other!

Parameterized Curve: ***Summary***

- Used for idealized to describe a structural change - i.e. a discontinuous component.
- Difficulty with parameterized curve descriptions if the DAE index is changing conditionally (= number of strings is changing)
- Currently not know how to translate automatically into a form which can be evaluated in a numerically sound way
- Friction elements always have a conditional index change (different number of moving elements)
- ***Pragmatic solution:*** **Transform manually** into form which can be handled
- ***General advice:*** for mechanical models, introduce discontinuities on force/acceleration (not velocity or position)