

# Modelica

## Diskrétní a hybridní systémy

jezekfi1@fel.cvut.cz

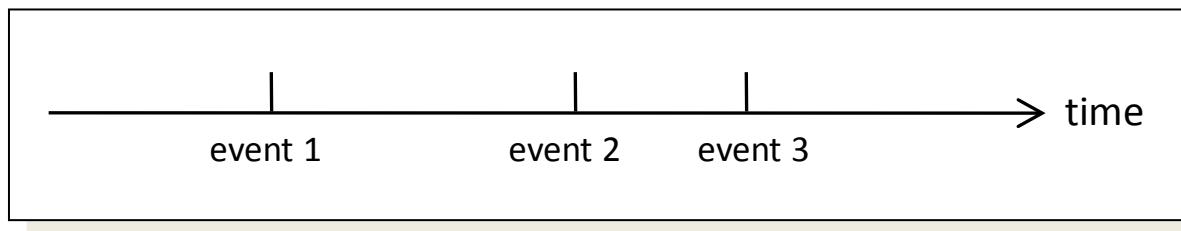
*Použity části z přednášek Petera Fritzsona Discrete Events and Hybrid Systems*

# Do teď jsme probírali spojité modely

- Co je to spojitý model? Co je to spojitý systém?
- V přírodě (makrosvět) většinou spojitý
- Potřebujeme zjednušovat

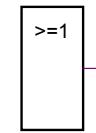
# Diskrétní systémy

- Změna stavu pouze v „clock“
- V mezičase setrvalé



# Digitální systémy

- Modelica.Electric.Digital
  - Hradla, RS, D
  - Logic, nikoli Boolean!
  - Logic hodnota;



## Logic value

'U'

## Meaning

Uninitialized

'X'

Forcing Unknown

'0'

Forcing 0

'1'

Forcing 1

'Z'

High Impedance

'W'

Weak Unknown

'L'

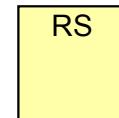
Weak 0

'H'

Weak 1

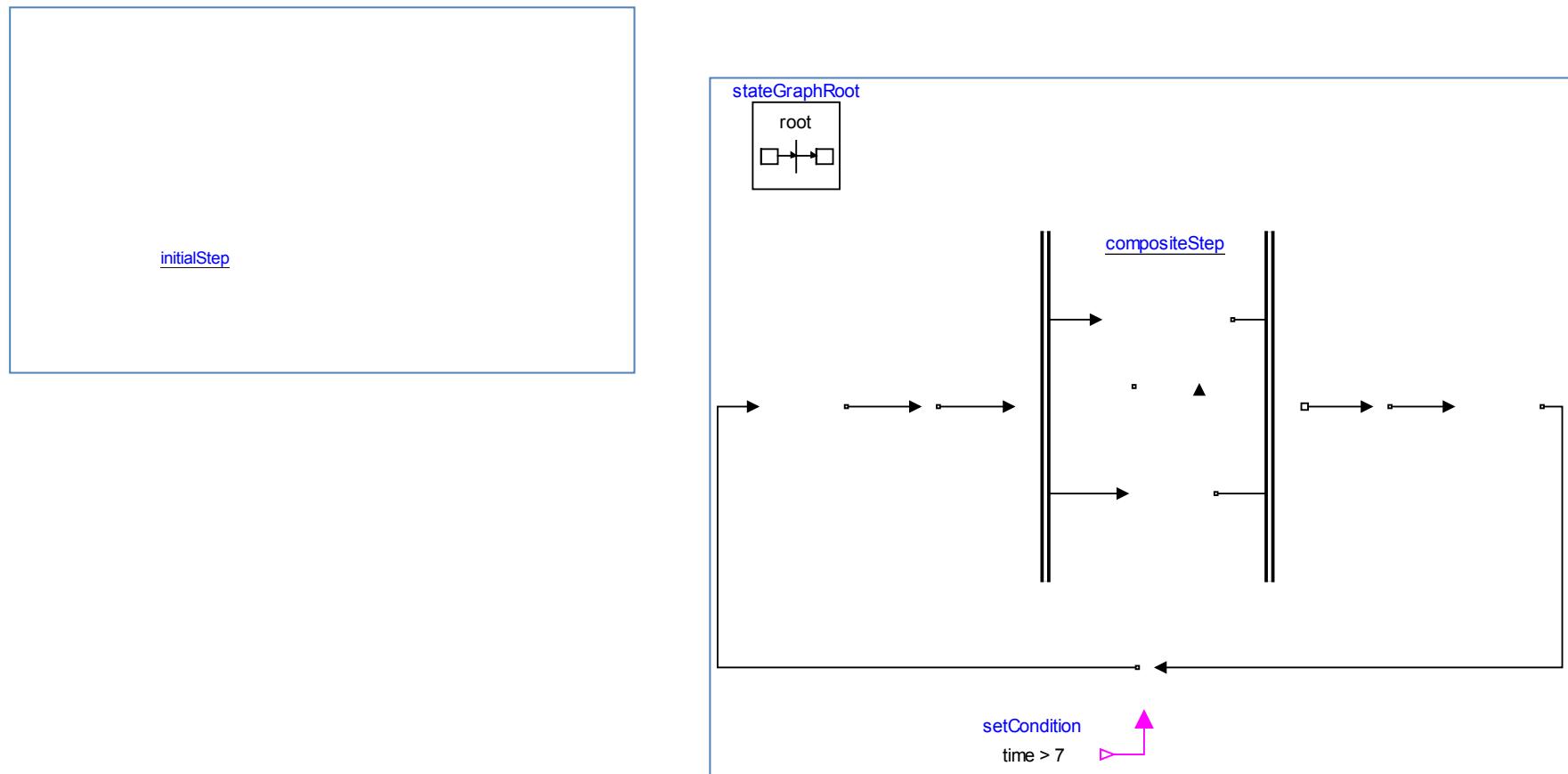
'\_'

Don't care



# Modelica.StateGraph

- Pro simulace stavových automatů



# Základní diskrétní typy

- Boolean
- Integer
- Discrete Real

Spojité

- Real

# Základ – if-equation

- If-equation
  - if  $x > 5$  then
    - $k = \text{true};$
    - $x = \text{whatever};$
  - elseif
    - ...
  - else
    - $k = \text{false};$
    - $x = \text{whatever2};$
  - end;
- Pozor:
  - Stejný počet rovnic ve VŠECH větvích
  - Vždy musí být **else** (pokud se jednoznačně nevylučuje)

# If-expressions

- If-expression

Boolean returning;  
equation

returning = if velocity < 0.0 then true else false;

- nebo

Boolean returning;  
equation

returning = velocity < 0.0;

- Cílový překlad

- Dá se zapsat také jako if-equation:

Boolean returning;  
equation  
if velocity < 0.0 then  
    returning = true;  
else  
    returning = false;  
end if;

# Event creation – if

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

```
if <condition> then  
  <equations>  
elseif <condition> then  
  <equations>  
else  
  <equations>  
end if;
```

```
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;
```

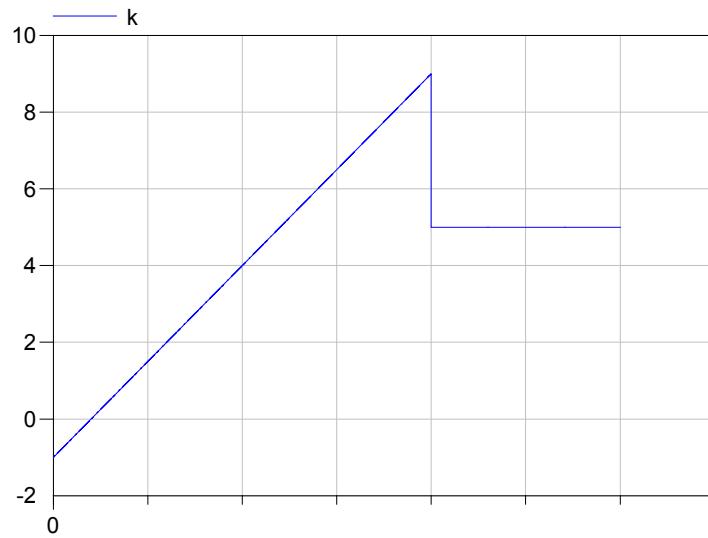
False if  $s < 0$

If-equation choosing  
equation for  $v$

If-expression

# IF

- Jakýkoli if-equation musí být rozepsatelný do if-expression
- Nemožno míchat s der
- Musí být pokryty všechny možnosti
- Postupuje se shora!



```
Real k;  
Real m;  
equation  
der(m) = 1;  
if time <= 20 then  
    k = 0.5*m - 1;  
elseif time < 1 then  
// ignorováno  
    k = 0;  
else  
    k = 5;  
end if;
```

# Time

- Časová pozice
  - time
    - s\_boolean = time > 30;
    - Je\_pozde = if time > 14.15 then true else false;
    - cas = time;
    - when time > 20. then
      - casZlomu = time;
    - end when;

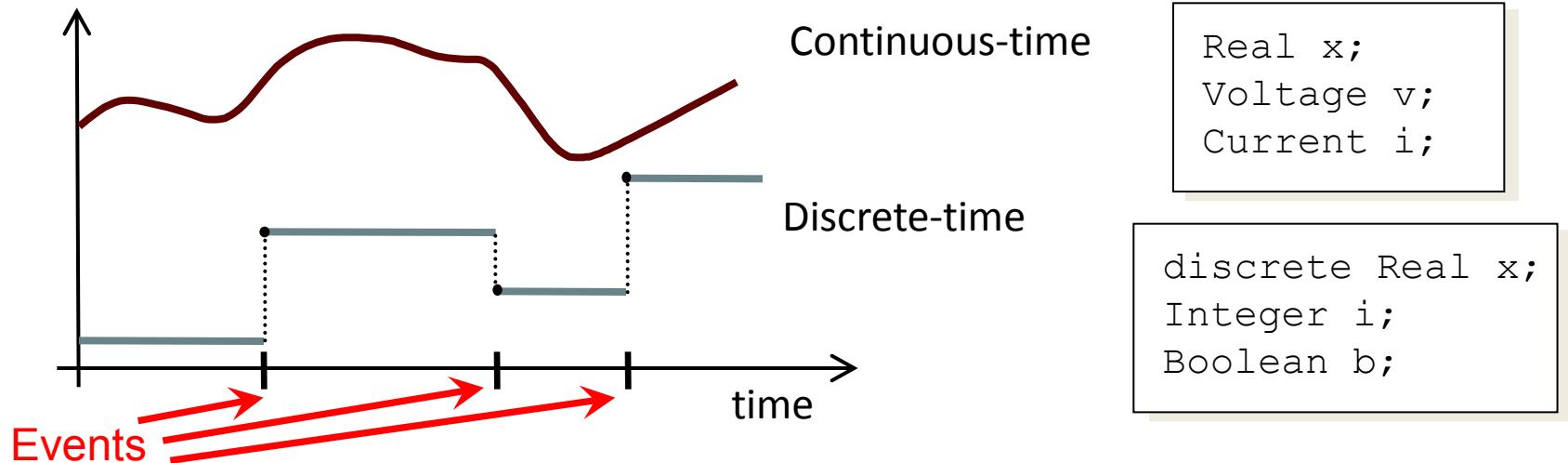
# Co to jsou EVENTS

- Události nespojitosti
- Zastaví se výpočet integrace, nalezne se přesný čas události
  - (Přesný ale není naprosto přesný)
  - Chyba: nelze hledat přesnou rovnost
    - Místo toho např.  $m > 3$  and  $m < 3 + 1e-6$
- Od času události platí rovnice v druhé větvi *if*

```
Real m;  
equation  
if m == 3 then  
    ...  
end if;
```

# 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

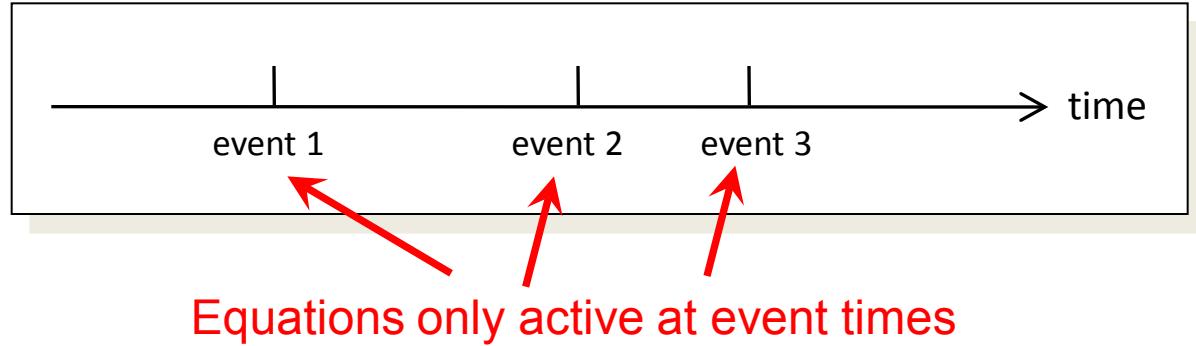
# When

- *If* rozděluje, KTERÉ rovnice použít
- Rovnice *when* se provedou POUZE při události
  - When se chová trochu jako algorithm
  - Máme stop-stav

# Event creation – when

*when-equations*

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



Time event

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

Only dependent on time, can be scheduled in advance

State event

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

Related to a state. Check for zero-crossing

# 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] // When A: Increase x by 1.5 at time=2
      x = pre(x)+1.5;
    end when;
    [when time>=1 then] // When B: Increase x by 1 at time=1
      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;
```

Pozor na Dymolu!

# Vnoření if a when

- Ify můžeme vnořovat (vždy lze vygenerovat if-expression)
- Wheny vnořovat nelze!
- Ify do whenů však lze

# Reinit

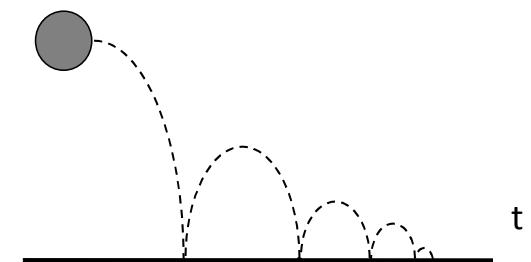
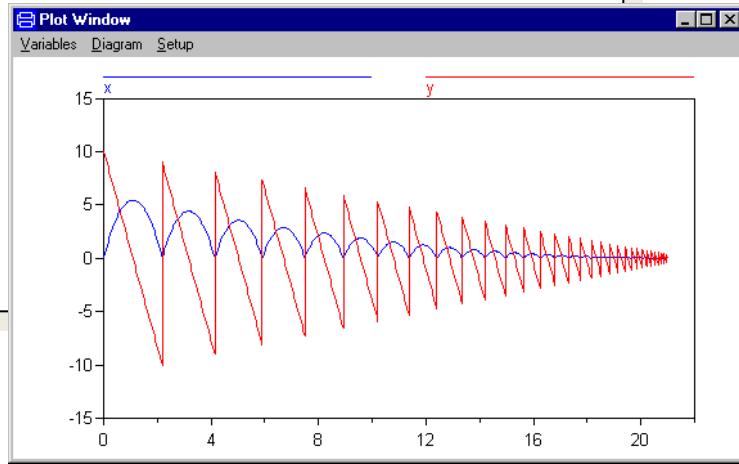
- Reinit nám dovolí měnit hodnoty spojité proměnné
- Lze použít pouze na proměnné, které se někde vyskytují v **der( )**
  - Jinak můžeme použít diskrétní proměnnou
- Pouze ve WHEN
  - reinit**(veličina, hodnota);
  - reinit**(rychlosť, 0.0);
- Bouncing ball

# 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;
```



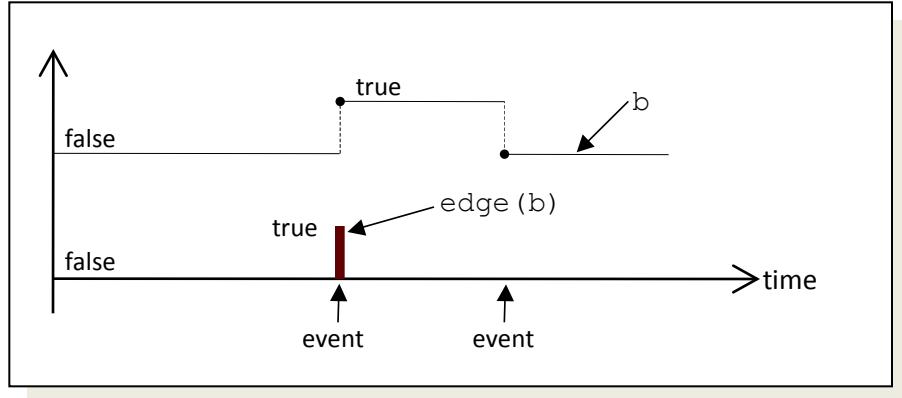
# Čím odpálíme eventy?

```
if (a > 3 and b < 1) or (C_boolean and D_integer == 4) then
```

- if/when time > 3.14159265358
  - if/when sample(start, interval)
  - if/when edge(x)
  - if/when change(x)
- 
- a\_bool = edge(x);
  - c\_boo = change(a\_boo);

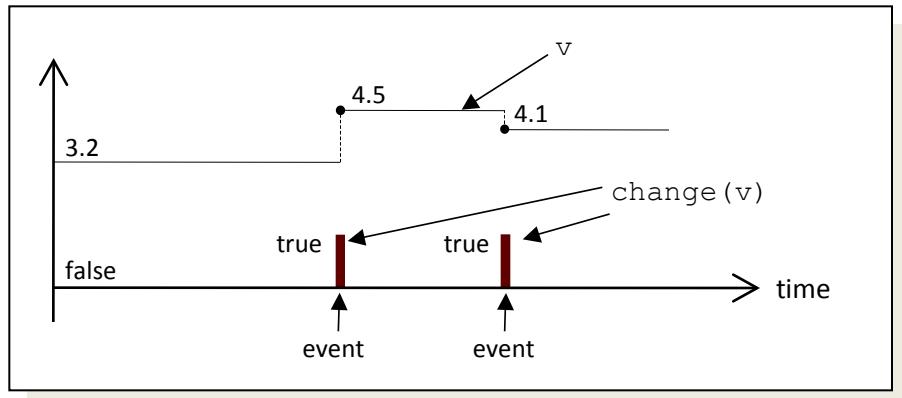
# Detecting Changes of Boolean Variables Using `edge()` and `change()`

Detecting changes of boolean variables using `edge()`



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

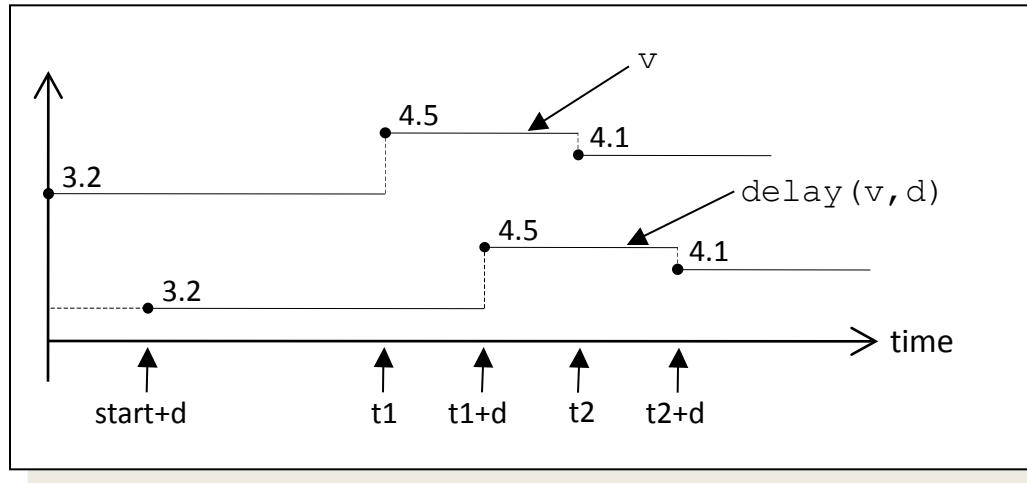
Detecting changes of discrete-time variables using `change()`



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

# Creating Time-Delayed Expressions

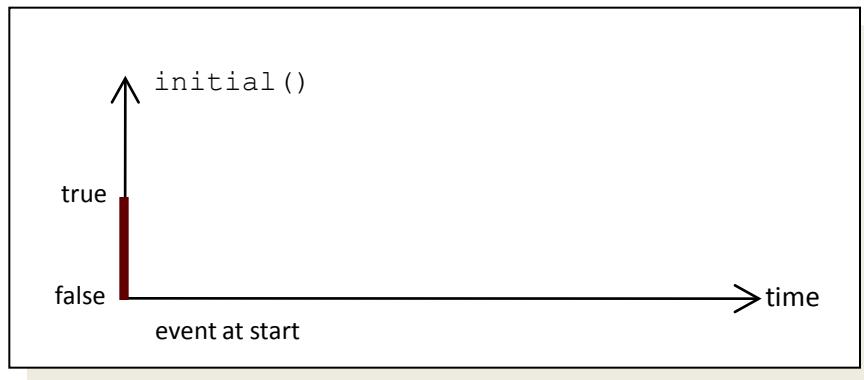
Creating time-delayed expressions using `delay()`



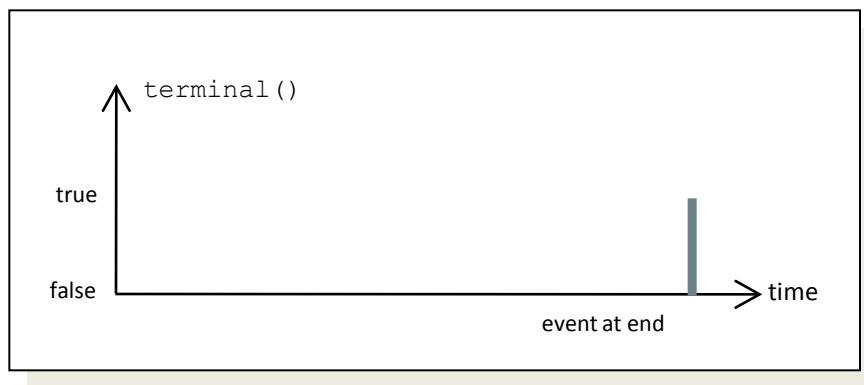
In the expression `delay(v, d)`  $v$  is delayed by a delay time  $d$

# initial and terminal events

Initialization actions are triggered by `initial ()`

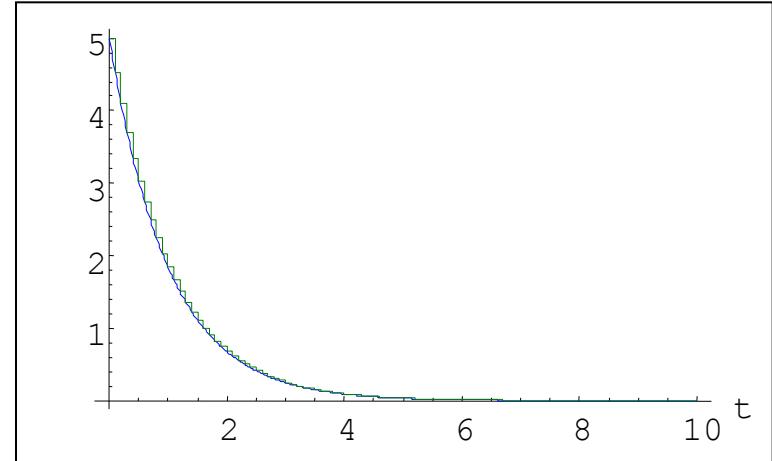


Actions at the end of a simulation are triggered by `terminal ()`



# A Sampler Model

```
model Sampler
  parameter Real sample_interval = 0.1;
  Real x(start=5);
  Real y;
equation
  der(x) = -x;
  when [sample](0, sample_interval) then
    y = x;
  end when;
end Sampler;
```

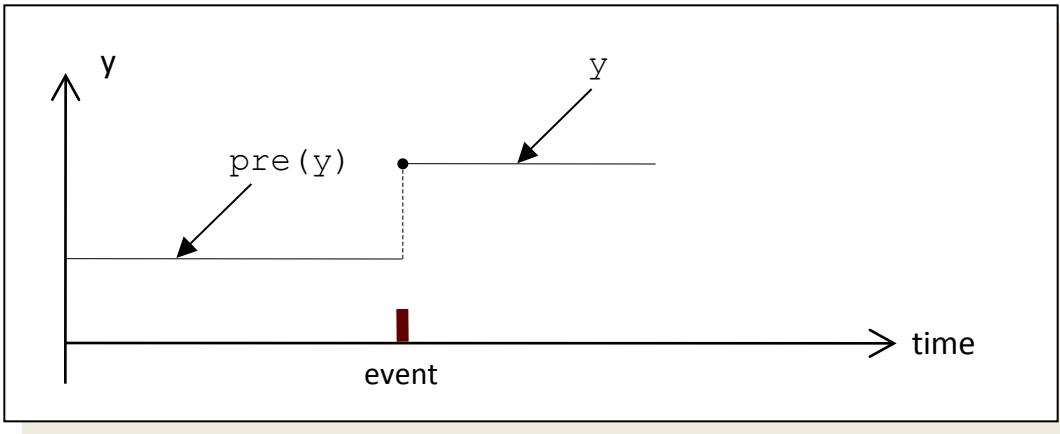


# PRE

- Předešlá hodnota
- ~ jiná proměnná

# Obtaining Predecessor Values of a Variable Using `pre()`

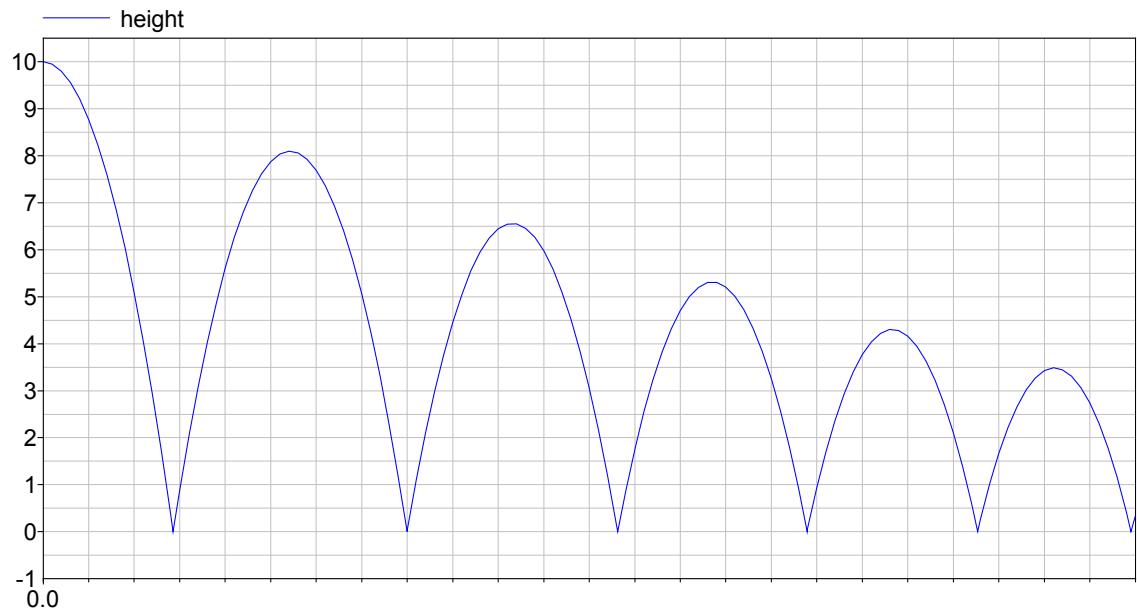
At an event, `pre(y)` gives the previous value of  $y$  immediately before the event, except for event iteration of multiple events at the same point in time when the value is from the previous iteration



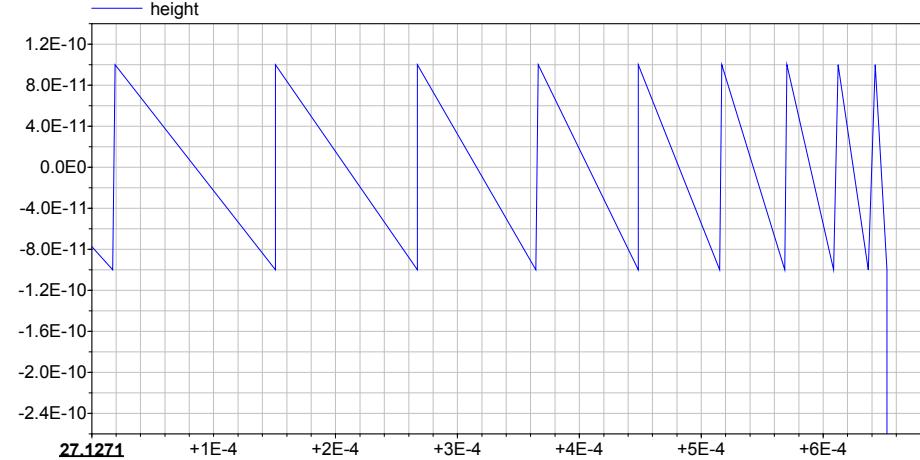
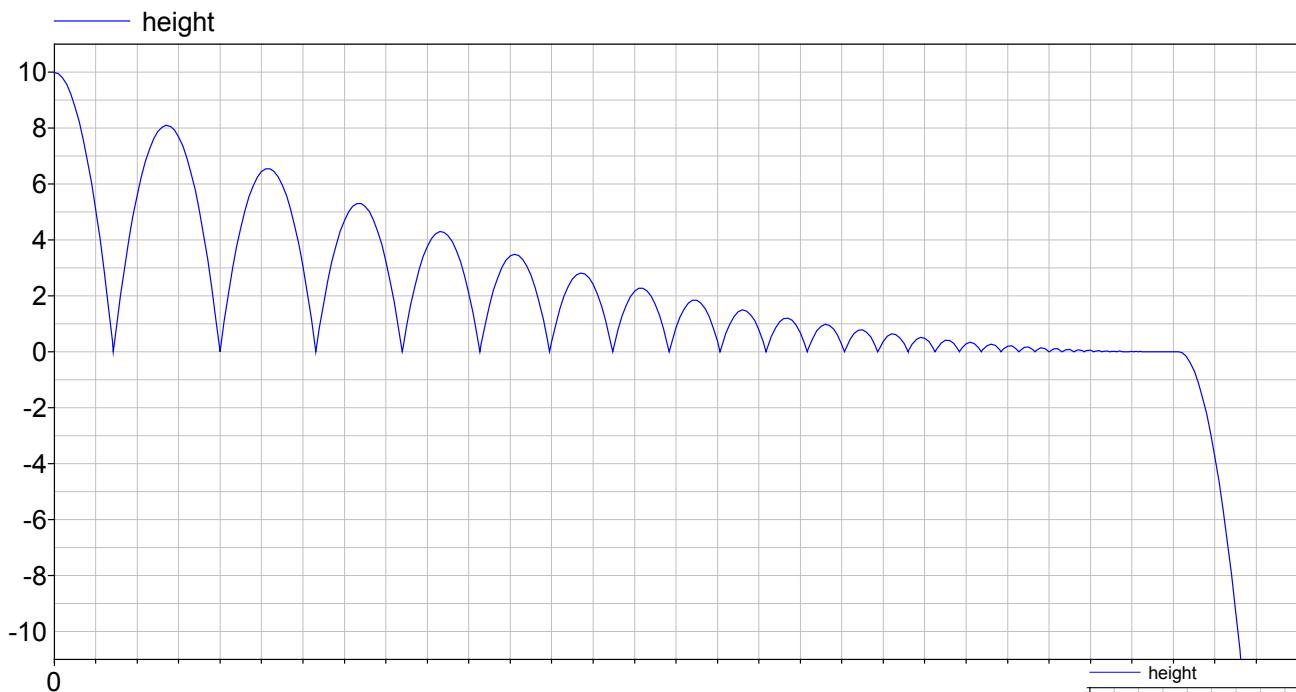
- The variable  $y$  has one of the basic types Boolean, Integer, Real, String, or enumeration, a subtype of those, or an array type of one of those basic types or subtypes
- The variable  $y$  is a discrete-time variable
- The `pre` operator can *not* be used within a function

# Zeno effect – bouncing ball - 1

```
model BouncingBall "the bouncing ball model - crash at T=27.5"
parameter Real g=9.81; //gravitational acc.
parameter Real c=0.90; //elasticity constant
Real height(start=10);
Real velocity(           start=0);
equation
  der(height) = velocity;
  der(velocity)= -g;
  when height<0 then
    reinit(velocity, -c*velocity);
  end when;
end BouncingBall;
```



# Zeno effect – bouncing ball - 2



# Zeno effect – bouncing ball - 3

- Řešení
  - Dát si na to pozor
  - Terminate simulation

```
when height < 1e-6 then
    terminate("Propadli sme podlahou");
end when;
```
  - Rozlišit Letí / stojí

# Terminating a Simulation

The `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:

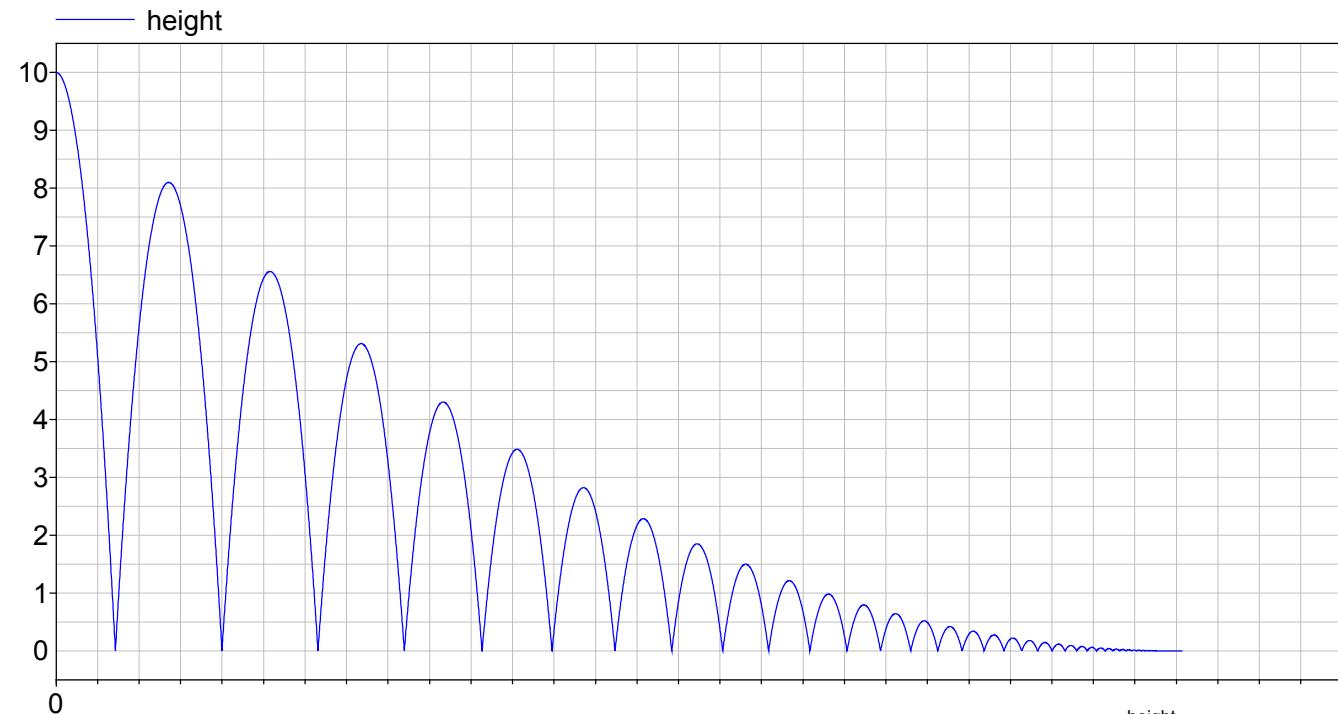
```
model terminationModel
  Real y;
equation
  y = time;
  when y > 5 then
    terminate("The time has elapsed 5s");
  end when;
end terminationModel;
```

terminate

Simulation ends before  
reaching time 10

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

# Zeno effect – bouncing ball - 4



# Dioda



—

# Dioda

- Jednoduše:
- Zavíráme a otvíráme jen proudem?
- Pozor na nulu!

```
if i < 0 then  
    v = Roff*i;  
else  
    v = Ron*i;  
end if;  
  
If v < 0 then  
    v = Roff*i;  
else  
    v = Ron*i;  
end if;
```

# Dioda 2

- Pomocí vložené proměnné

$\text{off} = s < 0;$

**if off then**

$v = s;$

**else**

$v = 0;$

**end if;**

$i = \text{if off then } 0 \text{ else } s;$

- Anebo lépe

$\text{off} = s < 0;$

//off – s je napeti, not off – s je proud

$v = s * (\text{if off then } 1 \text{ else } R_{on});$

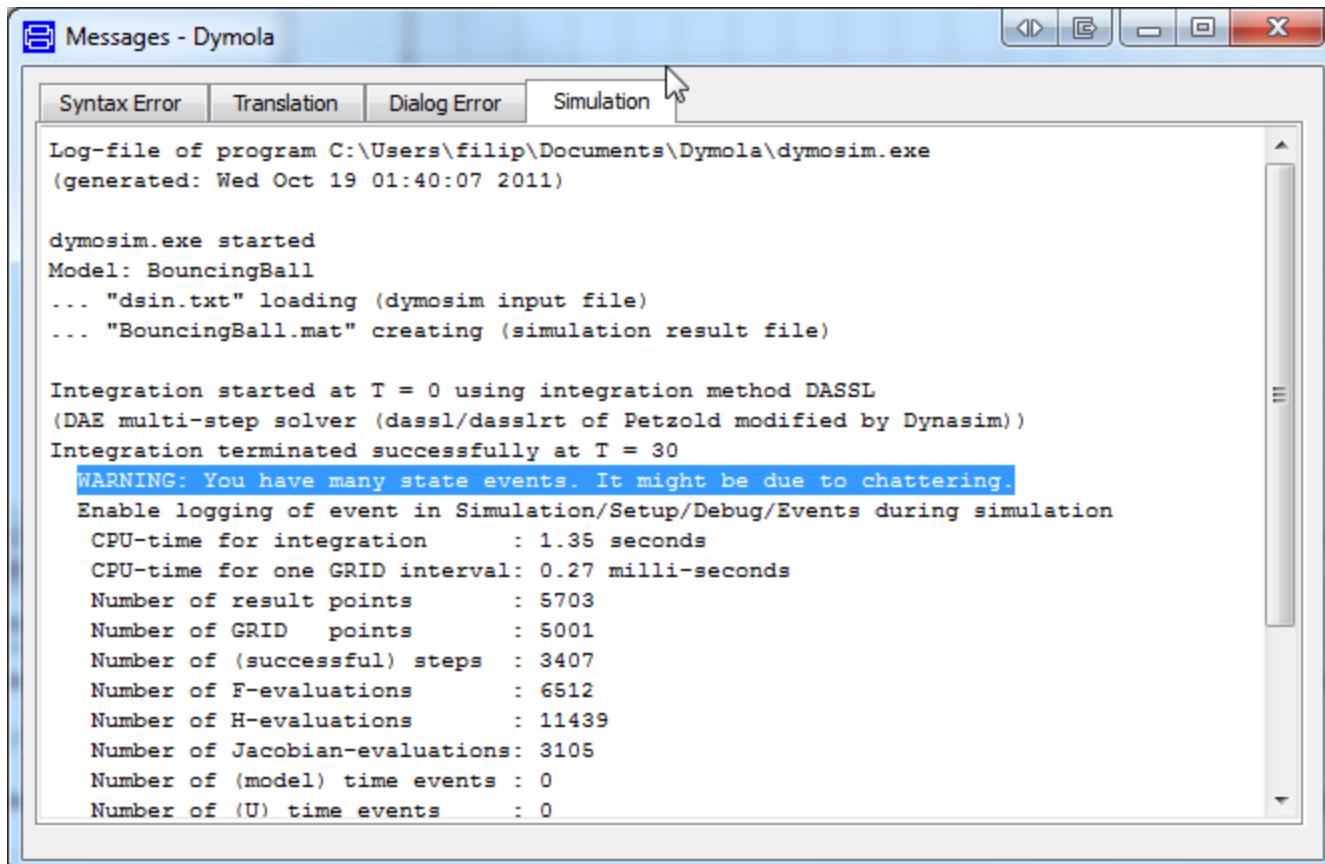
$i = s * (\text{if off then } 1/R_{off} \text{ else } 1);$

# Modelica.Electrical.Analog.Ideal.IdealDiode

```
constant Modelica.SIunits.Voltage unitVoltage= 1 annotation(HideResult=true);
constant Modelica.SIunits.Current unitCurrent= 1 annotation(HideResult=true);
equation
  off = s < 0;
  v = (s*unitCurrent)*(if off then 1 else Ron) + Vknee;
  i = (s*unitVoltage)*(if off then Goff else 1) + Goff*Vknee;
```

# Možné problémy

- Chattering – někde nám to příliš kmitá
- Numerika, numerika, numerika..



The screenshot shows a Windows application window titled "Messages - Dymola". The window has a standard title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with tabs: "Syntax Error", "Translation", "Dialog Error", and "Simulation". The "Simulation" tab is currently selected and highlighted in blue. The main area of the window contains text output from a simulation log. The log starts with the path and generation date of the log file, followed by the start of the simulation, loading of input files, and creation of result files. It then details the integration process, stating it started at T = 0 using DASSL and terminated successfully at T = 30. A yellow box highlights a warning message: "WARNING: You have many state events. It might be due to chattering." The log concludes with various performance metrics such as CPU time, number of evaluations, and event counts.

```
Log-file of program C:\Users\filip\Documents\Dymola\dymosim.exe  
(generated: Wed Oct 19 01:40:07 2011)

dymosim.exe started
Model: BouncingBall
... "dsin.txt" loading (dymosim input file)
... "BouncingBall.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 30
WARNING: You have many state events. It might be due to chattering.
Enable logging of event in Simulation/Setup/Debug/Events during simulation
CPU-time for integration      : 1.35 seconds
CPU-time for one GRID interval: 0.27 milli-seconds
Number of result points       : 5703
Number of GRID points         : 5001
Number of (successful) steps   : 3407
Number of F-evaluations       : 6512
Number of H-evaluations       : 11439
Number of Jacobian-evaluations: 3105
Number of (model) time events : 0
Number of (U) time events     : 0
```

# Příklady!!

# Shrnutí

- Co to je hybridní simulace
- Rozdíl mezi if a when
- Reinit
- Numerické problémy
- terminate()