Physical model based systems design

Albert Benveniste (Inria-Rennes)

Updated October 2016

Contents

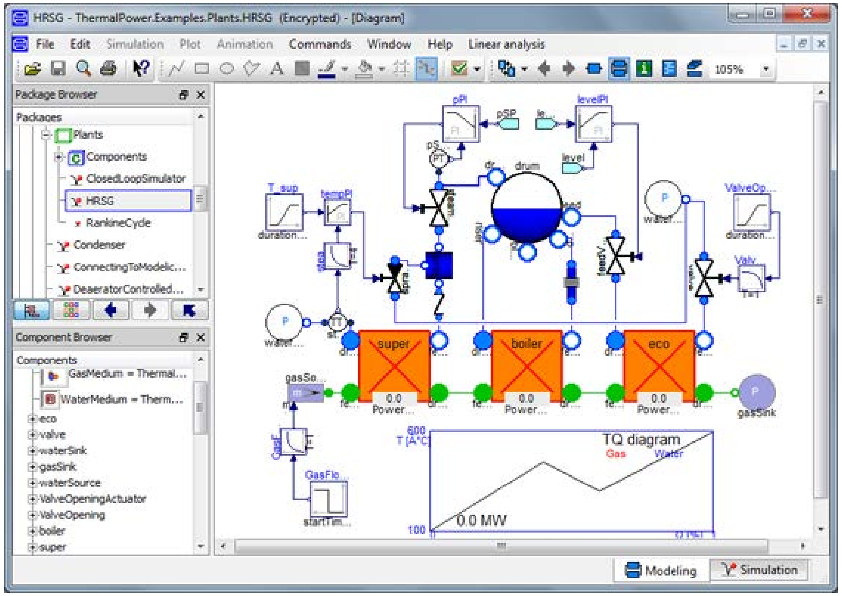
Acknowledgements:

- Benoît Caillaud, Khalil Ghorbal
- Tim Bourke, Marc Pouzet;
- Hilding Elmqvist, Martin Otter, Sven-Erik Mattsson;
- Paul Caspi and Paul Le Guernic .

Physical Modeling & Systems Design: the overall vision

Physical modeling using Modelica

(DAE: Differential Algebraic Equation)

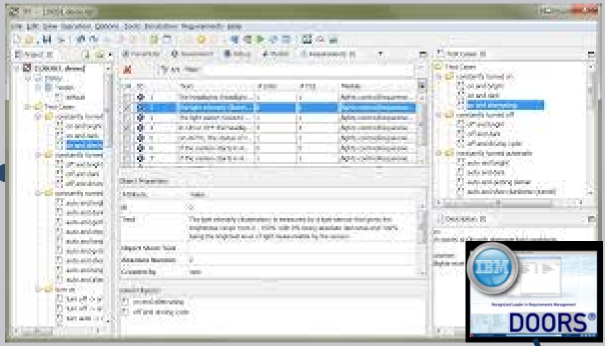


Modelica: multi-mode DAE systems:

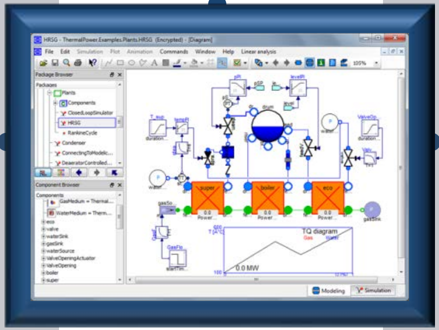
$$\begin{cases} \textbf{if } b \textbf{ do } F(\dot{x},x,u)=0 \\ b \;=\; BoolCond(\dot{x},x,u) \end{cases}$$

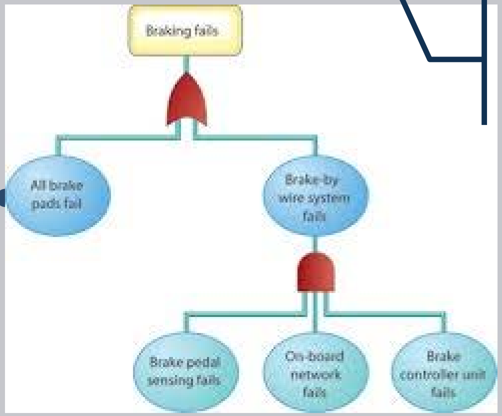
- **Modelica supports component based physical system modeling** (not Simulink)
- **Compilation is complex:**
  - Latent constraints
  - Index reduction
  - Structural analysis
- **Requires sophisticated causality analyses (as for bond graphs)**

The overall vision



Not discussed
see [Elmqvist 2014,2015]





Requir.

3D CAD

Safety Monitoring

Our focus

Modelica & Requirement Engineering

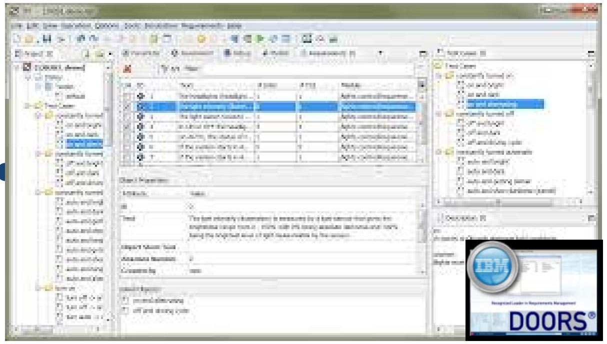
Albert Benveniste

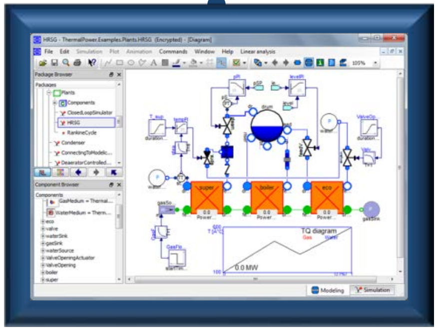
July 2014

Modelica + Requirements

Is this all we need? No!

Requirement architectures differ from (physical) system architectures





A requirement profile has been defined for Modelica [Fritzson14]

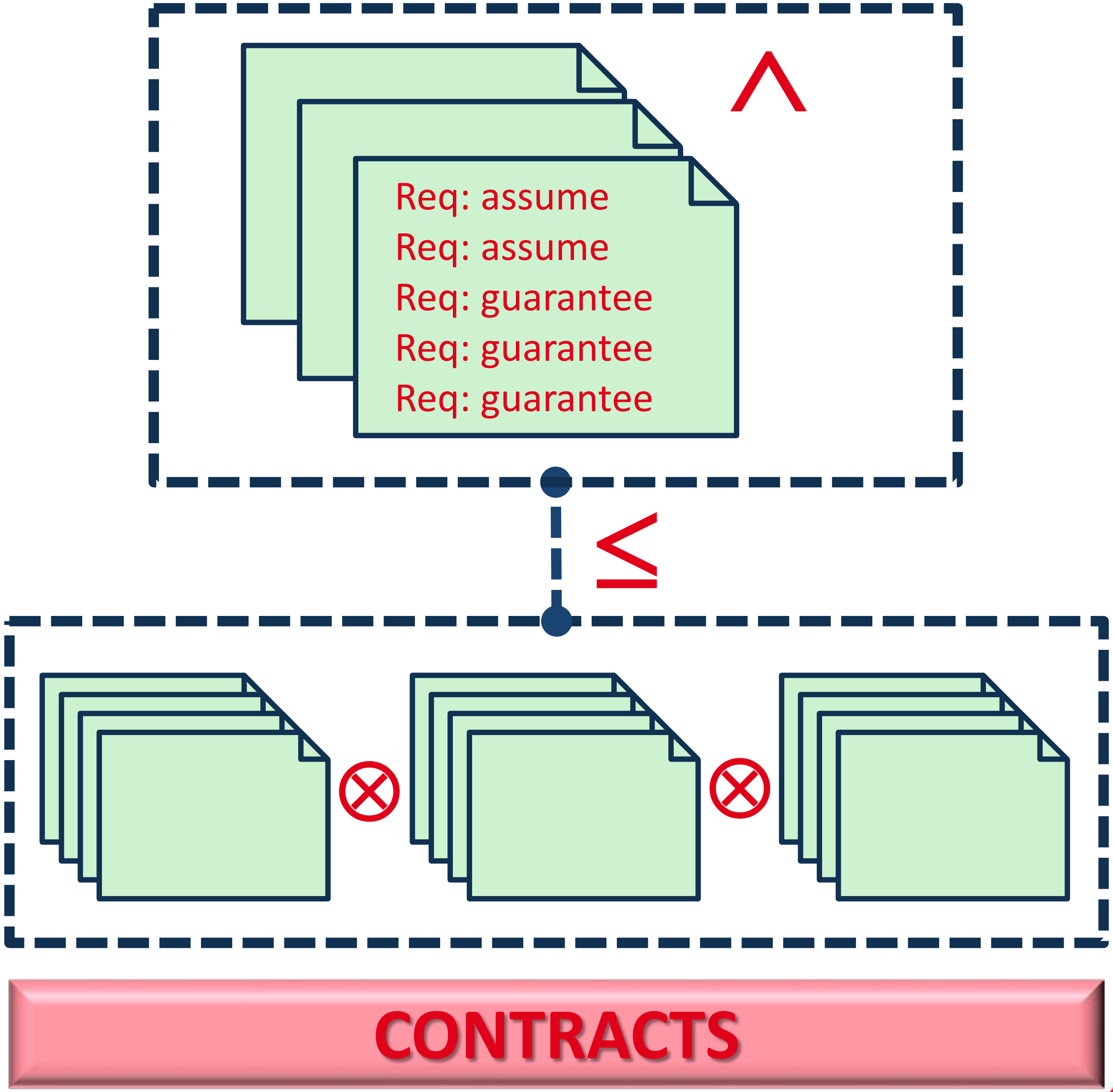
- Provision for writing requirements
- Linking requirements to test cases
- The link is syntactic, not semantic

Modelica + Requirements

Is this all we need? No!

Requirement architectures differ from (physical) system architectures

- Responsibilities must be clearly identified:
  - **guarantees**, vs.
  - **assumptions**
- **Conjunction** of requirements; several viewpoints
- From system to subsystem: **refinement** & **parallel composition**



Modelica + Contracts

- Assume/Guarantee contracts
- $C = (A, G) = (\textit{Assumption, Guarantee})$
  $= \text{pair of Modelica properties}$
- All the needed operators and relations exist
- Other forms of contract exist…

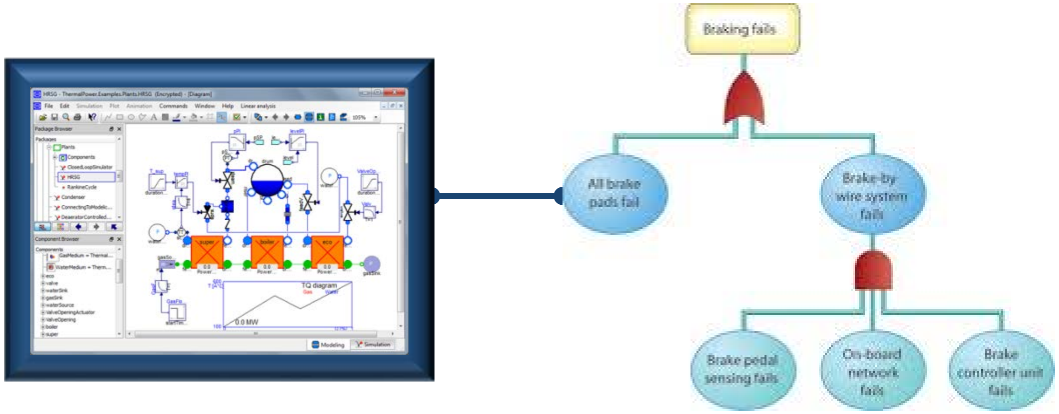
Modelica & Safety Engineering

Albert Benveniste

July 2014

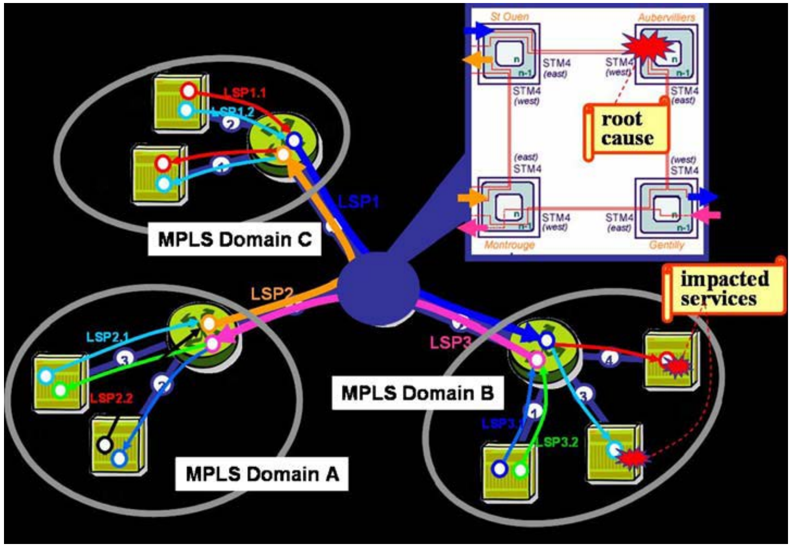
Modelica + Safety

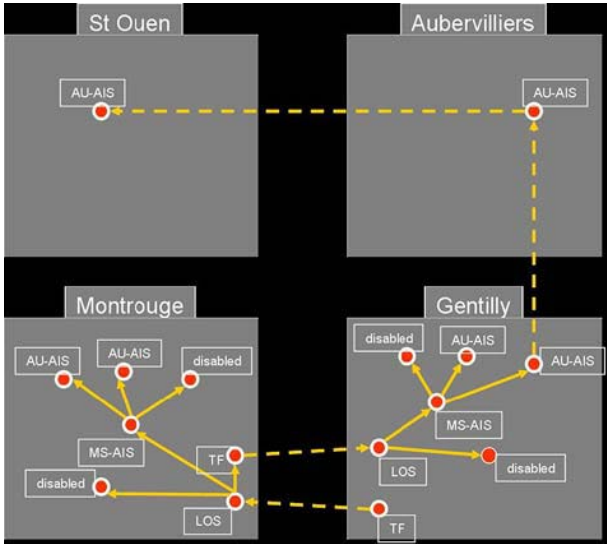
- Extend Modelica models with failure modes
- Use Modelica structural analysis to derive fault effects and propagation (fault tree)
- Check critical branches of the fault tree on the detailed Modelica model (guided simulation)

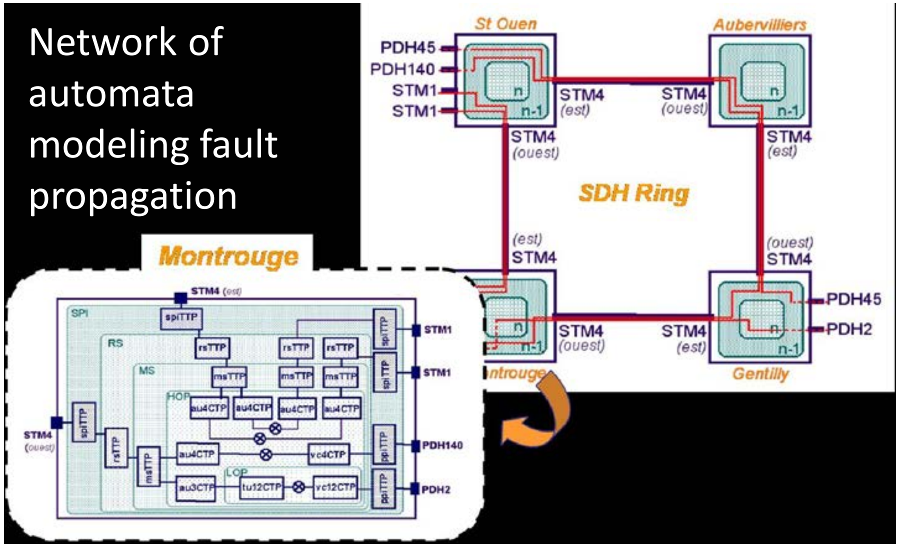


We can go beyond and perform system wide alarm handling

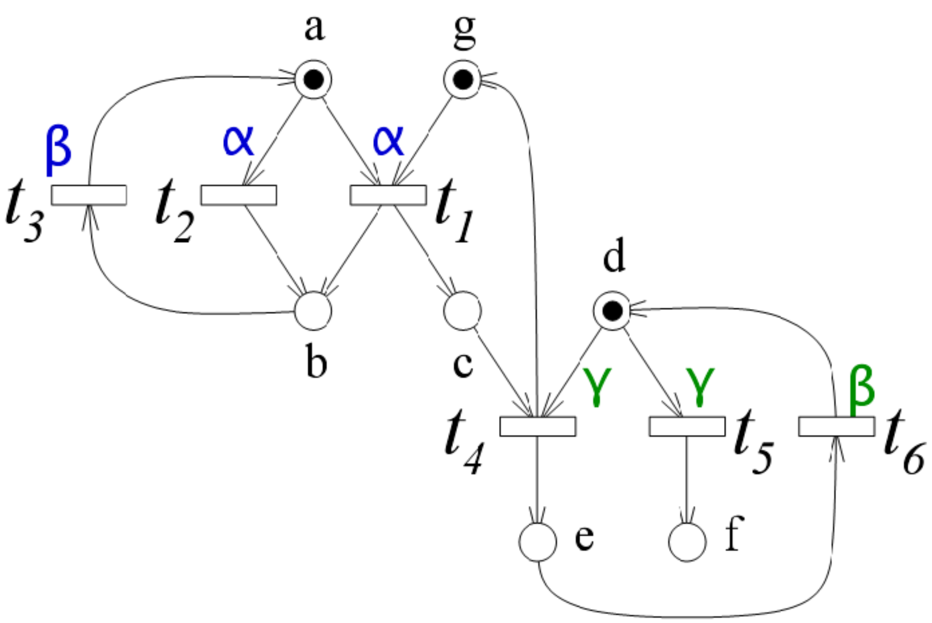
Telecom network diag [Fabre et al. 2006]







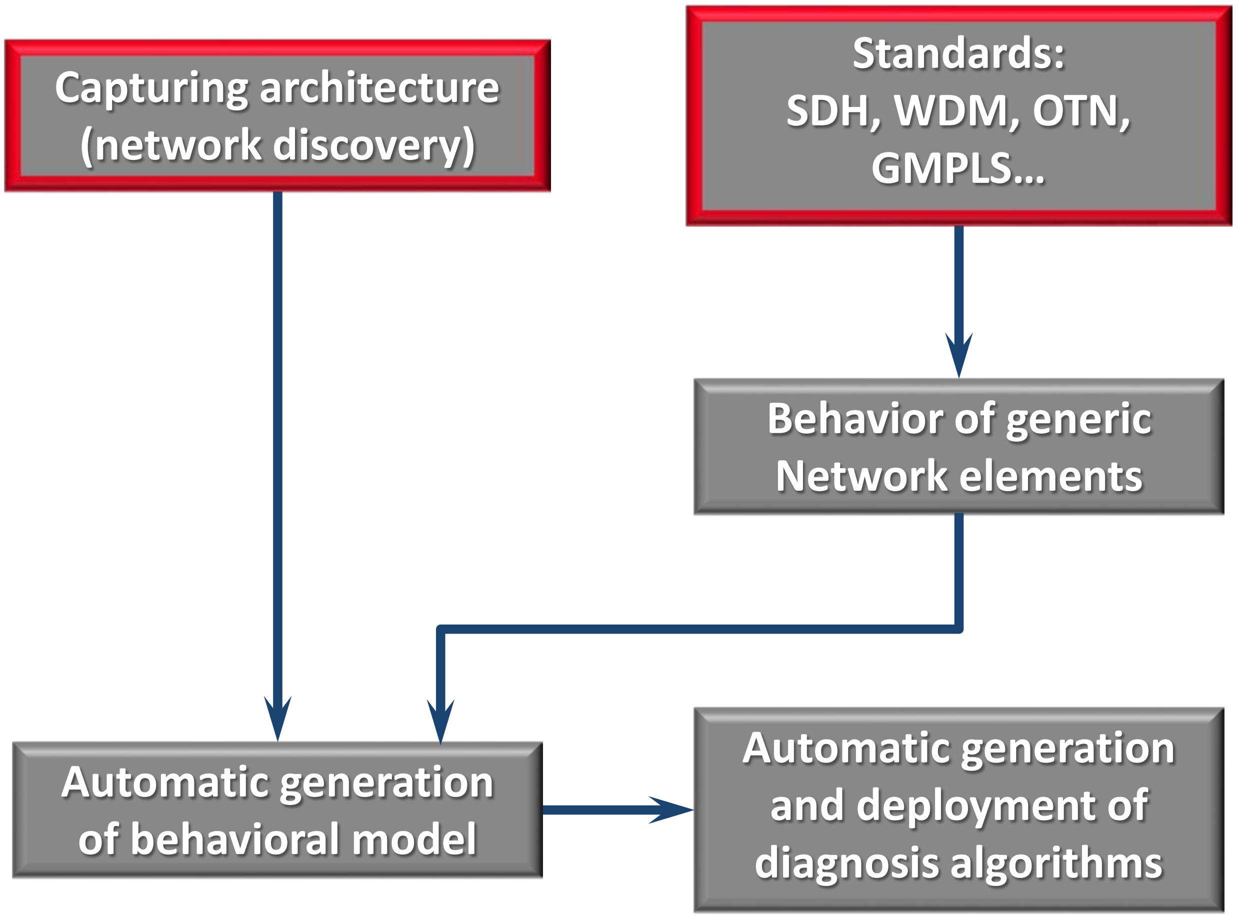
Diagnosis algorithm



- Automaton describing the operating modes (nominal, failed_x, …) and their transitions
- Some transitions are observed (alarms)
- System = product of many such automata
- **Reconstruct hidden state histories from observations (state observer)**

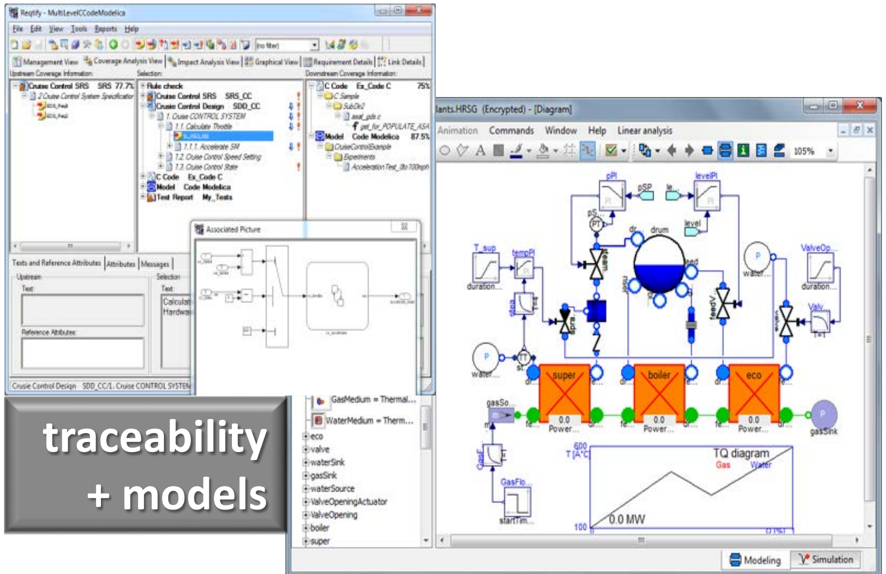
How to construct models?

Self-Modeling



How to construct models?

Self-Modeling



traceability
+ models

FMEA
(Fault Mode and
Effects Analysis)

Behavior of generic
elements

Automatic generation
of behavioral model

Automatic generation
and deployment of
diagnosis algorithms

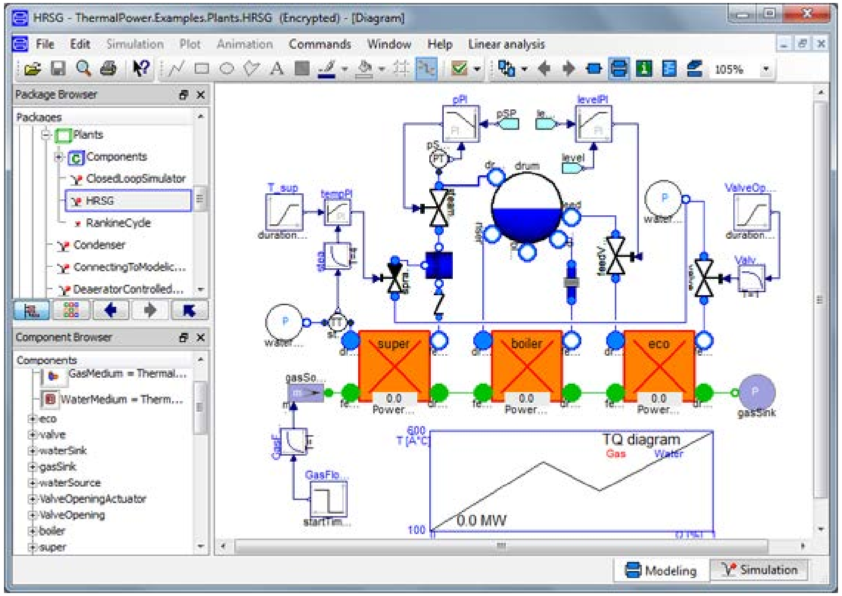
Modelica & System wide Diagnosis

Albert Benveniste

July 2014

Modelica

(DAE: Differential Algebraic Equation)

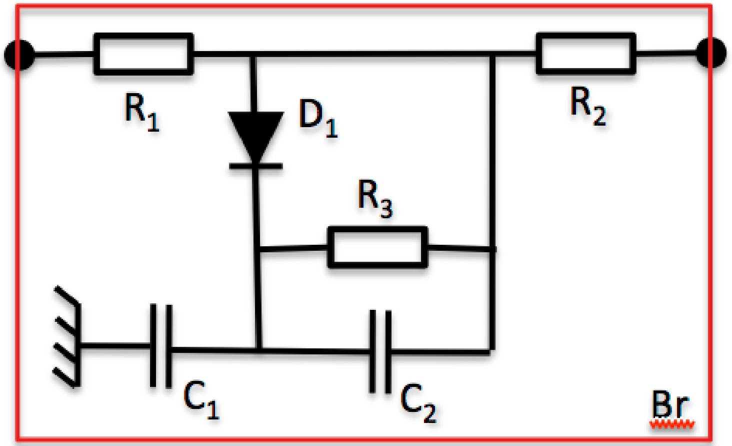


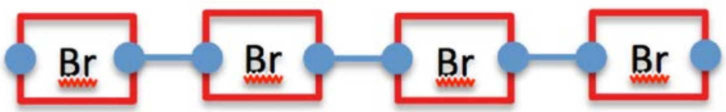
Modelica: multi-mode DAE systems:

$$\begin{cases} \textbf{if } b \ \textbf{ do } \ F(\dot{x}, x, u) = 0 \\ b \ = \ BoolCond(\dot{x}, x, u) \end{cases}$$

- **Modelica supports component based physical system modeling** (not Simulink)
- **Compilation is complex:**
  - Latent constraints
  - Index reduction
  - Structural analysis…
- **Requires sophisticated causality analyses (as for bond graphs)**
- **Idea: exploit the power of Modelica analyses by automatically deriving parity checks**

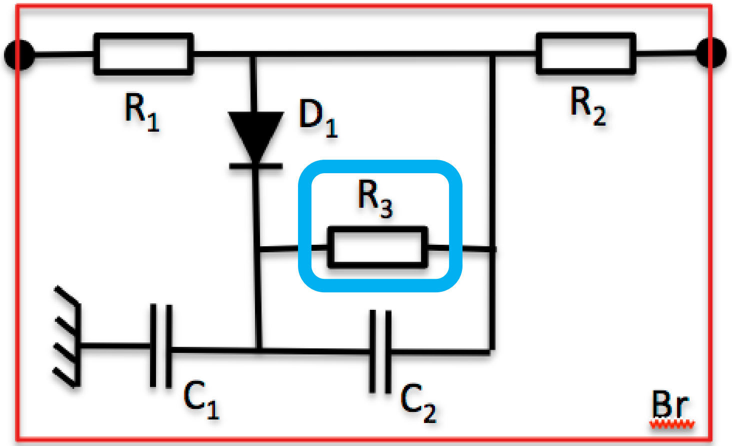
From Modelica to parity checks, automatically

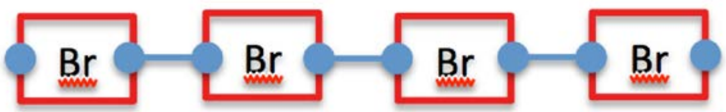




- Westinghouse braking system;
  control: pressure at the head of the train
- Each wagon induces two modes:
  valve D_1 open / closed
  - 2^n modes for a n wagons train
- Resistor R_3 captures possible leakage

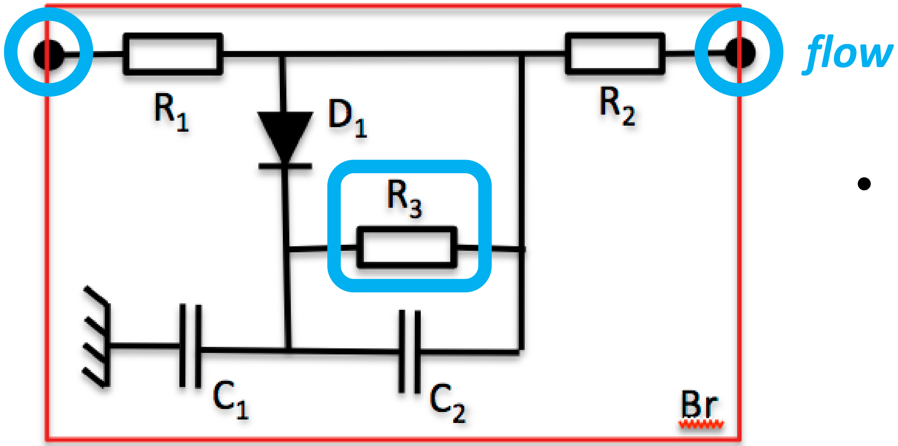
From Modelica to parity checks, automatically

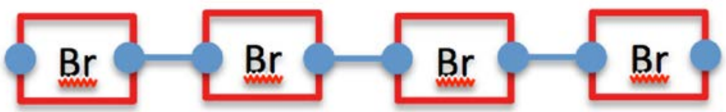




- Westinghouse braking system;
  control: pressure at the head of the train
- Each wagon induces two modes:
  valve D_1 open / closed
  - 2^n modes for a n wagons train
- Resistor R_3 captures possible leakage
  - Nominal / Leak : $R_3 = \infty$ / $R_3 < \infty$
- **Goal: monitoring for a possible leakage**
  - What should we measure?
  - Where to put sensors?
  - Getting all of this from Modelica compilation

From Modelica to parity checks, automatically

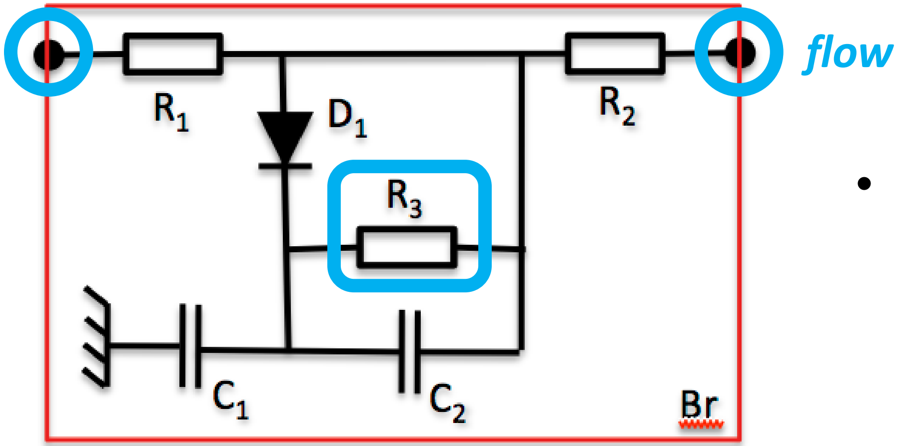


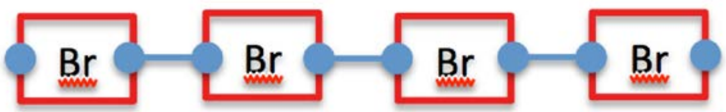


- The failure is non detectable when D_1 is open (no breaking mode)
  - (no flow traverses R_3 in this case)
  - Diagnosticability is mode-dependent (recall: 2^n modes for a n wagons train)

- How to generate parity checks
  - To monitor all possible leaks
  - By measuring (some or all of) the ***flows***?

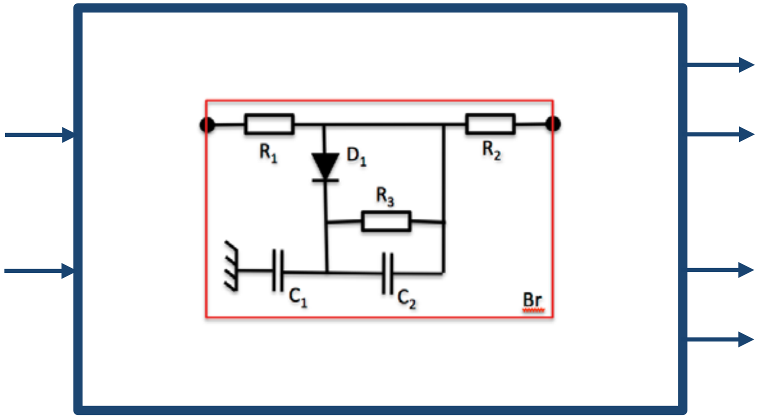
From Modelica to parity checks, automatically





- Idea: reuse the same Modelica model with the following adjustments:
  - Subset of the flows $\varphi_{j_1}, \ldots, \varphi_{j_k}$: inputs (possibly constrained)
  - Resistors $R_1, \ldots, R_n$: nominal parameters
  - Unobserved states $X = (x_1, \ldots, x_m)$
- The mode-dependent causality analysis of the Modelica model reveals that diagnosticability is mode-dependent

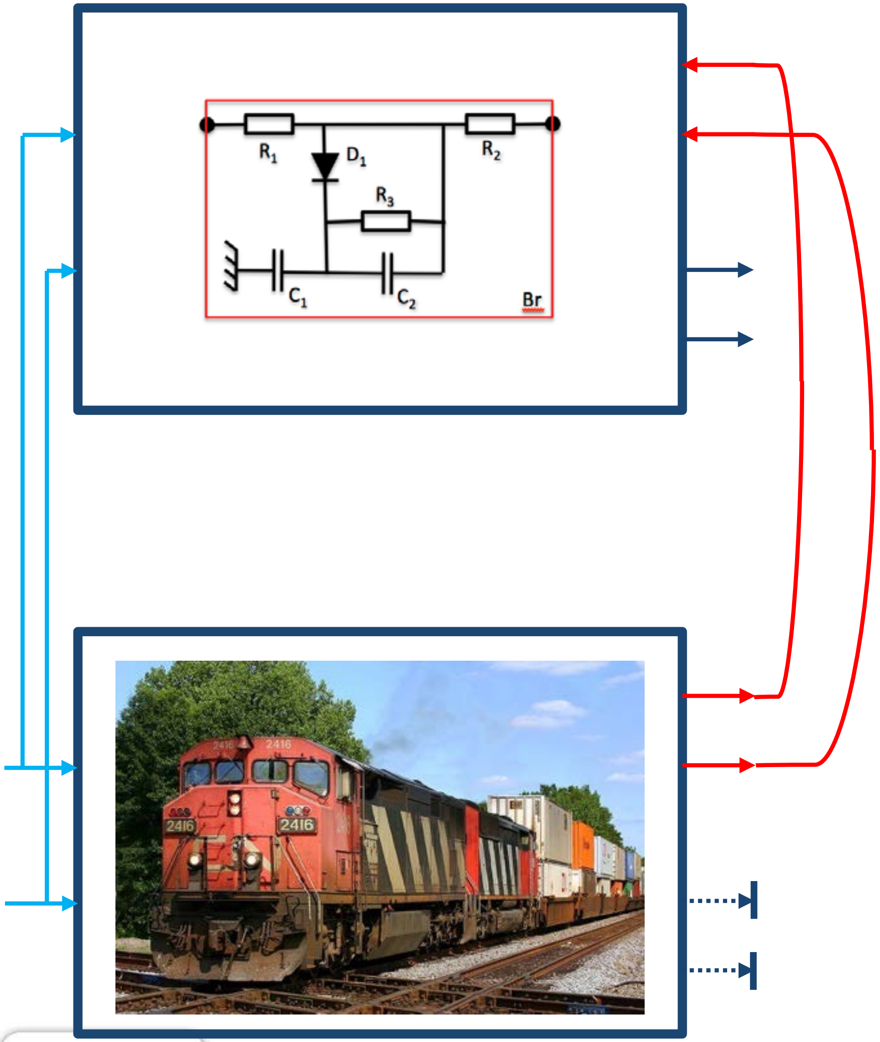
From Modelica to parity checks, automatically



- We have our Modelica model for simulation

- And the actual system for monitoring
- Some (but not all) states or outputs are measured

From Modelica to parity checks, automatically



- And feed the Modelica model with all the measurement data
- Yields an overconstrained Modelica model; exploit it to measure model/data fit

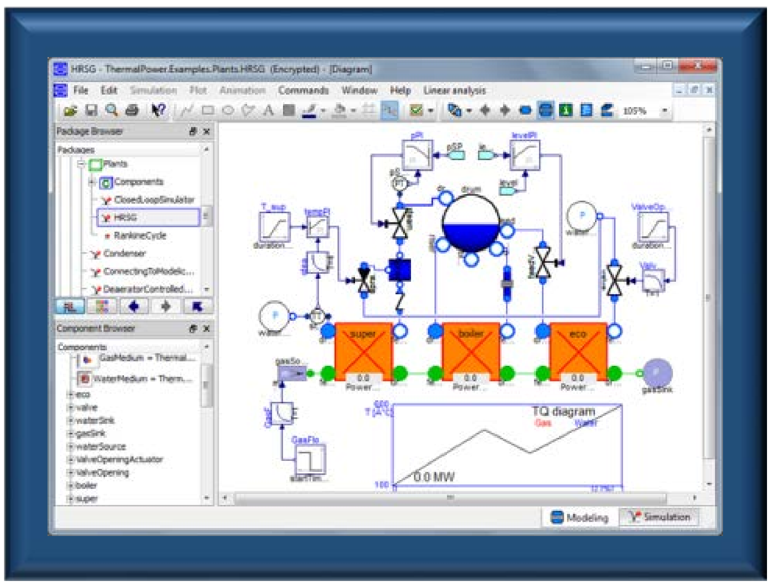
- Collect measurement data from the system in operation

Contents

1. Physical Modeling & Systems Design: a vision
2. The foundations for compiling Modelica (multi-mode DAE systems)

The need for flexibility and solid foundations

Modelica, thou shall be flexible and formally sound



- **Flexible**
  - Simulating
  - Supporting safety analyses
  - Generating fault trees
  - Generating parity equations
  - Handling multi-mode with no restriction
  - Supporting non-regular systems?
- **Formally sound**
  - Benefiting from the heritage of synchronous languages

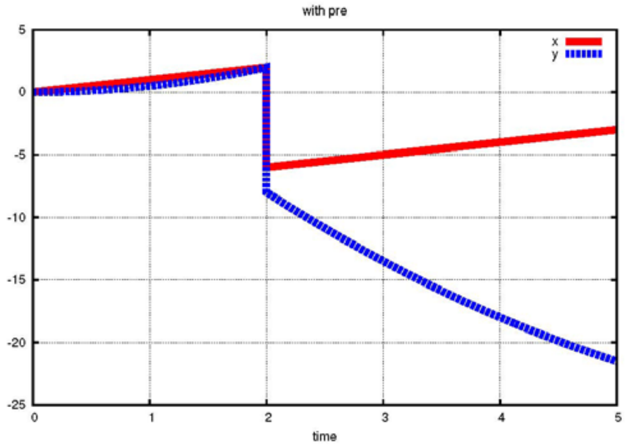
Challenging hybrid causal loops in Modelica tools

```
model scheduling
  Real x(start=0);
  Real y(start=0);

equation
  der(x)=1;
  der(y)=x;

  when x>=2 then
    reinit(x,-3*pre(y));
  end when;
  when x>=2 then
    reinit(y,-4*pre(x));
  end when;

end scheduling
```



At the instant of reset, x and y each have a value defined in terms of their values just prior to the reset.

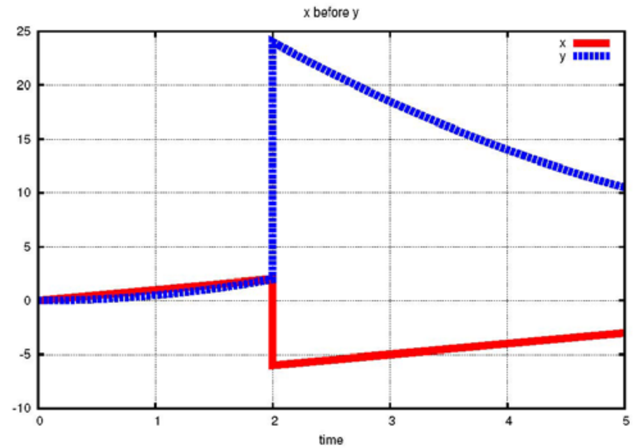
Challenging hybrid causal loops in Modelica tools

```
model scheduling
  Real x(start=0);
  Real y(start=0);

equation
  der(x)=1;
  der(y)=x;

  when x>=2 then
    reinit(x,-3*y);
  end when;
  when x>=2 then
    reinit(y,-4*x);
  end when;

end scheduling
```



Take the pre's away:

At the time of reset, x and y are in cyclic dependency chain.

The simulation runtime (of both OpenModelica and Dymola), chooses to reinitialize x first, with the value -6 as before, and then to reinitialize y in terms of the updated value of x: 24.

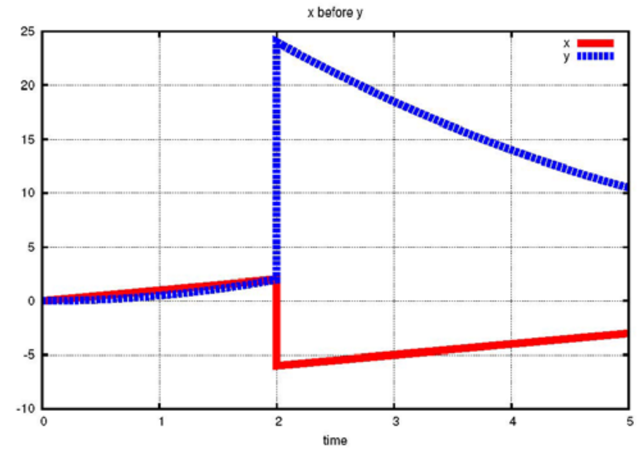
Challenging hybrid causal loops in Modelica tools

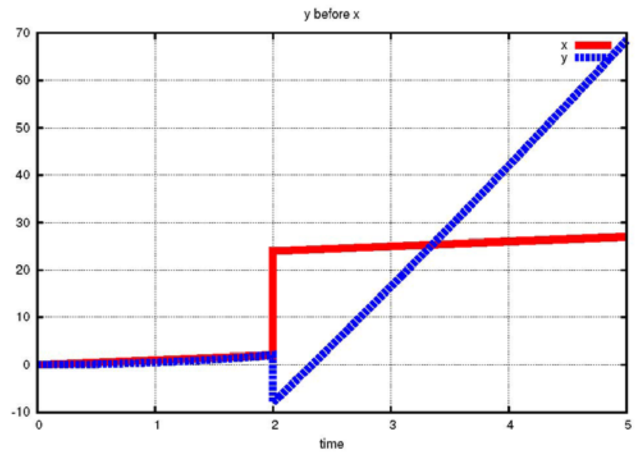
```
model scheduling
  Real x(start=0);
  Real y(start=0);

equation
  der(x)=1;
  der(y)=x;

  when x>=2 then
    reinit(x,-3*y);
  end when;
  when x>=2 then
    reinit(y,-4*x);
  end when;

end scheduling
```





What happens, if we reverse the order of the two reinit?...

The simulation result changes, as shown on the bottom diagram.

The same phenomenon occurs if the reinit's are each placed in their own when clause.

Challenging hybrid causal loops in Modelica tools

- The causal version (with the “pre”) is scheduled properly and simulates as expected.
- The non-causal programs are accepted as well, but the result is not satisfactory.
- Algebraic loops cannot be rejected, even in resets, since they are just another kind of equation. They should be accepted, but the semantics of a model must not depend on its layout!
- Studying causality can help to understand the detail of interactions between discrete and continuous code.

All about synchronous languages in a few slides

Compilation schemes from the Constructive Semantics

Albert Benveniste

July 2014

An example of Signal program and its compilation

- **Why discussing Signal?**
  - Among synchronous languages, Signal is closest to Modelica
  - It has clocks and equations on clocks, and
  - Requires mode-dependent causality analysis

An example of Signal program and its compilation

- **Why discussing Signal?**
  - Among synchronous languages, Signal is closest to Modelica
  - It has clocks and equations on clocks, and
  - Requires mode-dependent causality analysis

- **The Signal vintage watch**
  - This is an old mechanical watch like the one I have. Turn the button. The watch goes for some time, and then stops. When it stops, turn again the button... and so on...

An example of Signal program and its compilation

```
(  X := IN default ZX-1
| ZX := X$1 init 0
| IN ^= when (ZX ≤ 0) )
```

Input IN returns X

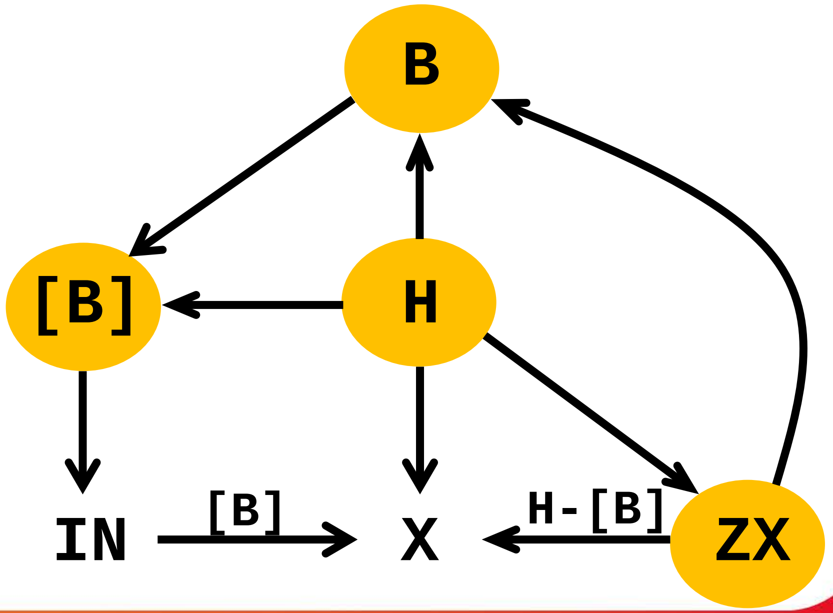
This was Signal code; Lustre-like pseudo-code follows:

```
pre(X) init 0 in
if pre(X) ≤ 0
      then (get IN and set X := IN)
      else (set X := pre(X)-1)
```

An example of Signal program and its compilation

```
(  X := IN default ZX-1
| ZX := X$1 init 0
|  B := (ZX ≤ 0)
| IN ^= (when B) ^< B
|  H ^= B ^= X ^= ZX )
```

[B]: when B

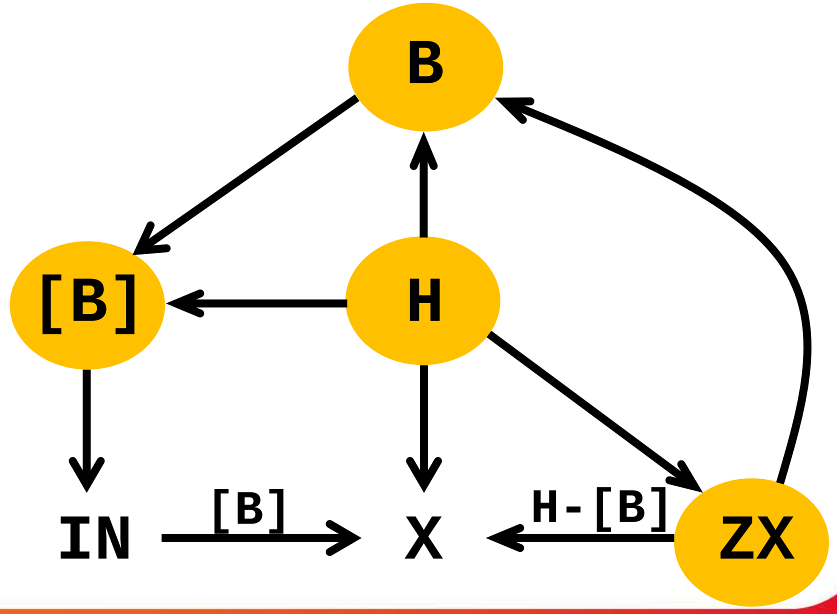


An example of Signal program and its compilation

```
(  X := IN default ZX-1
| ZX := X$1 init 0
|  B := (ZX ≤ 0)
| IN ^= (when B) ^< B
|  H ^= B ^= X ^= ZX )
```

```
[B]: when B
```

```
Case B = true
Case B = false
```

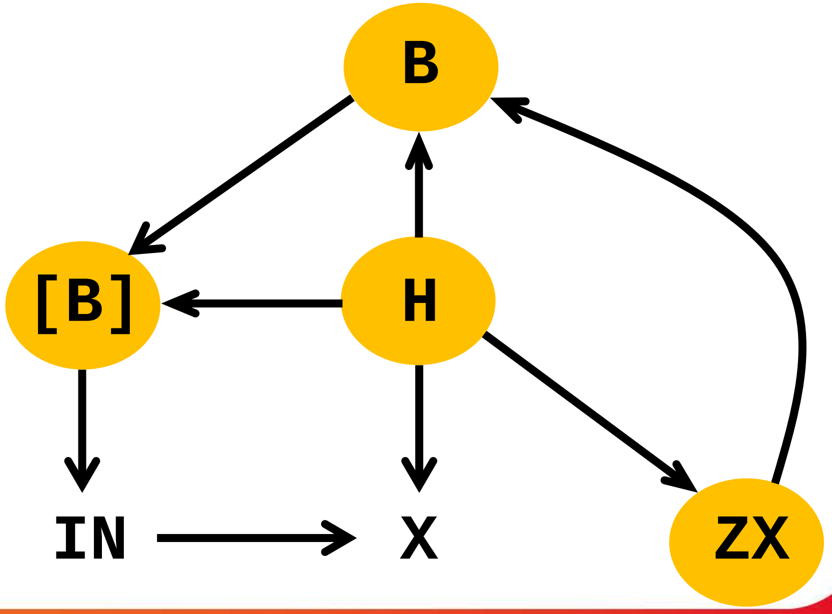


An example of Signal program and its compilation

```
(  X := IN default ZX-1
| ZX := X$1 init 0
|  B := (ZX ≤ 0)
| IN ^= (when B) ^< B
|  H ^= B ^= X ^= ZX )
```

`[B]: when B`

`Case B = true`

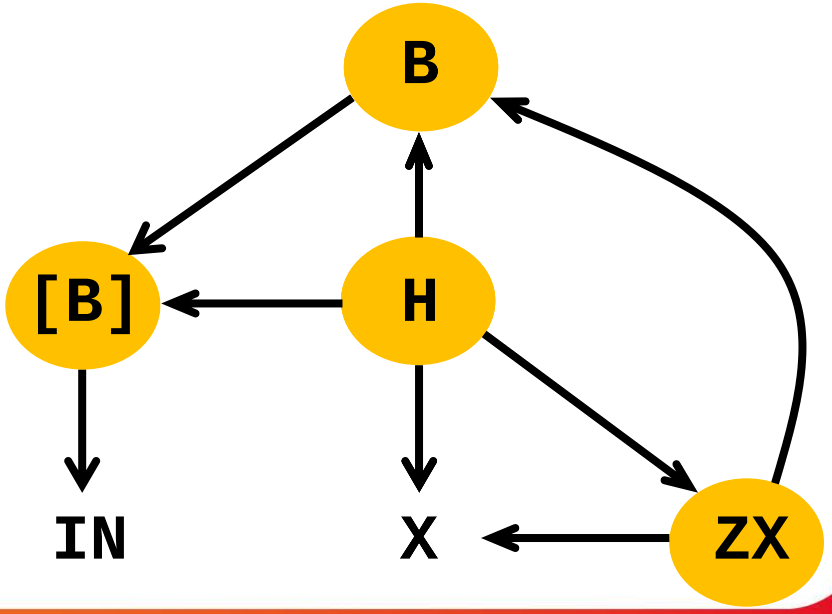


An example of Signal program and its compilation

```
(  X := IN default ZX-1
| ZX := X$1 init 0
|  B := (ZX ≤ 0)
| IN ^= when B ^< B
|  H ^= B ^= X ^= ZX )
```

```
[B]: when B
```

```
Case B = false
```



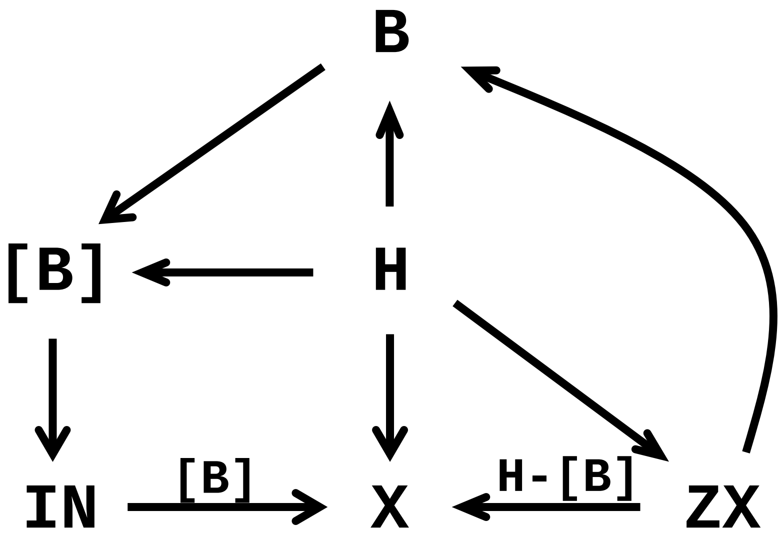
An example of Signal program and its compilation

Constructive Semantics:

execution scheme that schedules

atomic actions (here: evaluating expressions)

and successfully evaluates all variables at each reaction



From Synchronous Languages to the Structural Analysis of multi-mode DAE systems

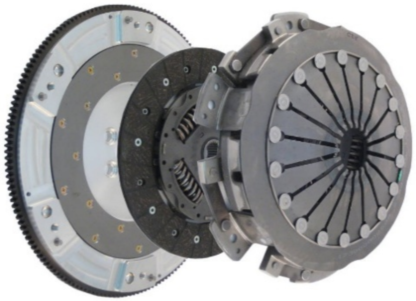
From continuous to discrete time using non-standard analysis

Albert Benveniste

July 2014

A simple clutch

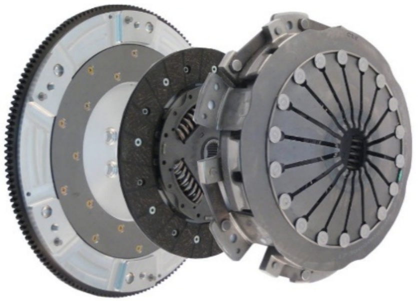
$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \text{ then } \begin{cases} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \\ \text{else } \begin{cases} \tau_1 = 0 \\ \tau_2 = 0 \end{cases} \end{cases}$$



- The clutch has two modes:
  - *engaged* : $\gamma = T$; DAE
  - *released* : $\gamma = F$; ODE

A simple clutch

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \text{ then} \begin{cases} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \\ \text{else} \begin{cases} \tau_1 = 0 \\ \tau_2 = 0 \end{cases} \end{cases}$$



- The clutch has two modes:
  - *engaged* : $\gamma = T$; DAE
  - *released* : $\gamma = F$; ODE
- Is it enough to make DAE analysis mode dependent?
  - problem: this says nothing about how to handle the resets at mode change
- When the clutch engages and the two rotation speeds differ, an impulse occurs for the torques
- This example is not supported by existing Modelica tools today

A simple clutch
the "engaged" mode

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \ \text{ then} \begin{cases} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \\ \text{else} \begin{cases} \tau_1 = 0 \\ \tau_2 = 0 \end{cases} \end{cases}$$

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' - \omega_2' = 0 & (e_4) \\ \tau_1 + \tau_2 = 0 & (e_5) \end{cases}$$

- The source DAE model is in black
- In red I have added a latent equation, which implicitly holds although not written in the source
- When all latent equations are added (here only 1), we inherit a structurally nonsingular system of algebraic eqns (e_1, e_2, e_4, e_5) with dependent variables $(\tau_1, \tau_2, \omega_1', \omega_2')$ (dummy derivatives)
- Solving it yields the velocities as an implicit function of the positions ≈ODE

A simple clutch trying existing tools

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \text{ then} \begin{cases} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \\ \text{else} \begin{cases} \tau_1 = 0 \\ \tau_2 = 0 \end{cases} \end{cases}$$

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' - \omega_2' = 0 & (e_4) \\ \tau_1 + \tau_2 = 0 & (e_5) \end{cases}$$

- Unfortunately, this tells nothing about how to handle the mode changes
- The difficult case is $\gamma : F \to T$ (the clutch gets engaged)
- Some simulation results for this example by existing tools follow

A simple clutch trying Modelica

```
model ClutchBasic
parameter Real w01=1;
parameter Real w02=1.5;
parameter Real j1=1;
parameter Real j2=2;
parameter Real k1=0.01;
parameter Real k2=0.0125;
parameter Real t1=5;
parameter Real t2=7;
Real t(start=0, fixed=true);
Boolean g(start=false);
Real w1(start = w01, fixed=true);
Real w2(start = w02, fixed=true);
Real f1;
Real f2;
equation
der(t) = 1;
g = (t >= t1) and (t <= t2);
j1*der(w1) = -k1*w1 + f1;
j2*der(w2) = -k2*w2 + f2;
0 = if g then w1-w2 else f1;
f1 + f2 = 0;
end ClutchBasic;
```

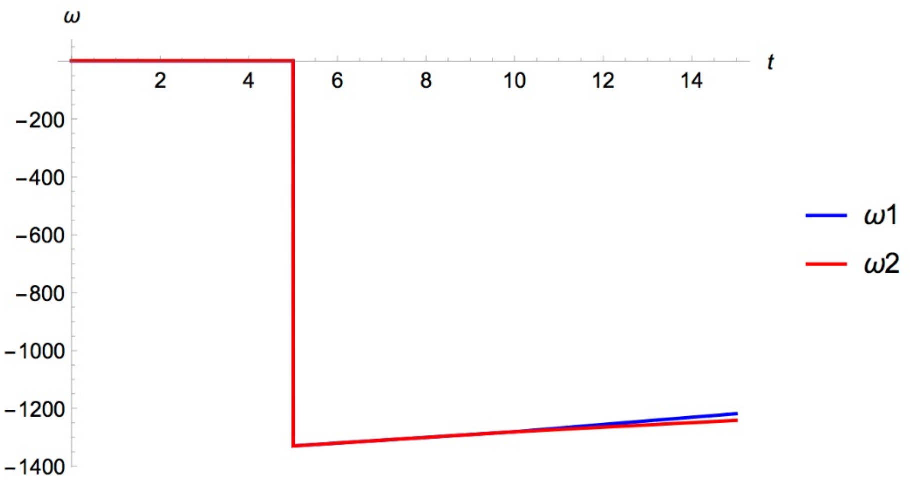
- Mode changes $F \to T \to F$ at $t = 5, 10$

```
The following error was detected at time: 5.002
Error: Singular inconsistent scalar system for
f1 = ((if g then w1-w2 else 0.0))/(-(if g then
0.0 else 1.0)) = -0.502621/-0
Integration terminated before reaching
"StopTime“ at T = 5
```

- The reason is that Dymola has symbolically pivoted the system of equations, independently of the mode. By doing so, it has produced an equation defining f1 that is singular in mode g.

A simple clutch trying Mathematica (NDSolve)

```
NDSolve[{
w1'[t] == -0.01 w1[t] + t1[t],
2 w2'[t] == -0.0125 w2[t] + t2[t],
t1[t] + t2[t] == 0,
s[t] (w1[t] - w2[t]) + (1 - s[t]) t1[t]
== 0,
w1[0] == 1.0, w2[0] == 1.501, s[0] == 0,
WhenEvent[t == 5,
{s[t] -> 1}
]
},
{w1, w2, t1, t2,s},
{t, 0, 7}, DiscreteVariables -> s]
```



No crash at mode change. But nondeterministic reset reveals that cold restart is indeed performed by NDSolve on this example.

Mode changes
$F \rightarrow T$ at $t = 5$
$T \rightarrow F$ at $t = 10$

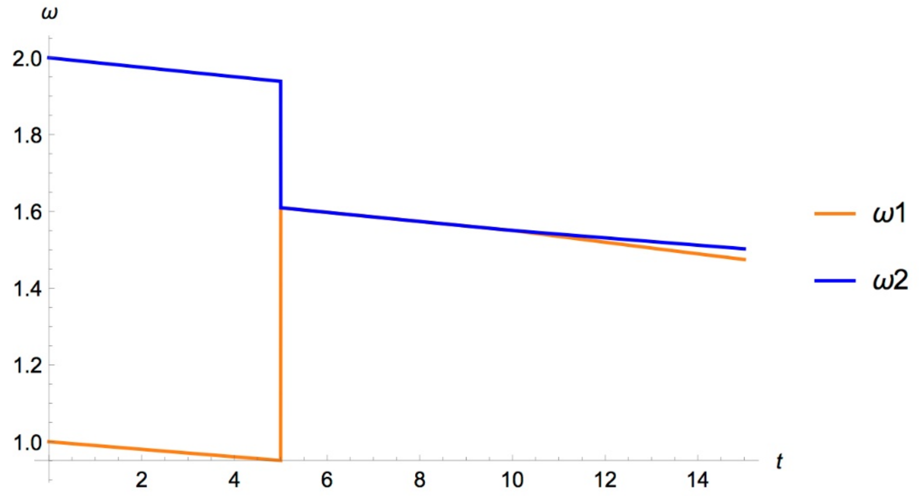
A simple clutch
a comprehensive approach

$$\left\{\begin{array}{l} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \text{ then} \left\{\begin{array}{l} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{array}\right. \\ \text{else} \left\{\begin{array}{l} \tau_1 = 0 \\ \tau_2 = 0 \end{array}\right. \end{array}\right.$$

- The difficult case is $\gamma : F \rightarrow T$ (the clutch gets engaged)
- We handle this by invoking nonstandard analysis and expand: $\omega' = \frac{\omega^{\blacksquare} - \omega}{\partial}$ where $\omega^{\blacksquare}$ is the "next" operator and time step ∂ is an infinitesimal of nonstandard analysis
- This brings the whole model to discrete-time and we are able to combine the techniques from synchronous languages with those of index analysis from DAE

A simple clutch our results

$$\begin{cases} \omega_1' = f_1(\omega_1, \tau_1) \\ \omega_2' = f_2(\omega_2, \tau_2) \\ \text{if } \gamma \text{ then} \begin{cases} \omega_1 - \omega_2 = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \\ \text{else} \begin{cases} \tau_1 = 0 \\ \tau_2 = 0 \end{cases} \end{cases}$$



The reset is handled satisfactorily. The rotation speed right after engagement sits between the two rotation speeds before, which matches the intuition from physics.

Mode changes
$F \to T$ at $t = 5$
$T \to F$ at $t = 10$

Structural Analysis of multi-mode DAE systems

See my detailed lecture

Albert Benveniste

July 2014

Conclusion

1. Physical modeling is central to systems design
   - Modeling for simulation
   - Modeling fault propagation
   - Generating parity checks for diagnostics
   - Complemented with modeling the computing architecture

Conclusion

1. Physical modeling is central to systems design
   - Modeling for simulation
   - Modeling fault propagation
   - Generating parity checks for diagnostics
   - Complemented with modeling the computing architecture

2. The compilation of physical models requires a difficult structural analysis
   - Source of difficulties in current tools
   - Techniques from synchronous languages help
   - Efficient algorithms are yet to obtain

Thanks

INVENTEURS DU MONDE NUMÉRIQUE