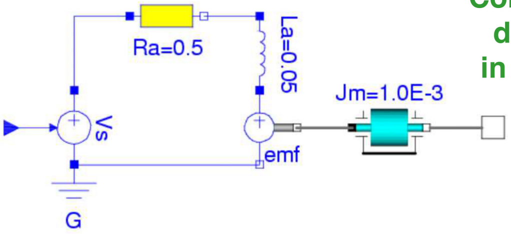
Index Theory and Structural Analysis for multi-mode DAE Systems

Albert Benveniste Benoît Caillaud Khalil Ghorbal (Inria, Rennes)
Marc Pouzet (ENS, Paris)
Hilding Elmqvist (Mogram, Lund) Martin Otter (DRL, Munich)

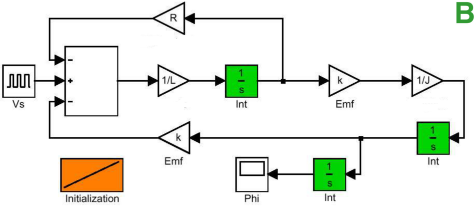
December 6, 2016

Compositionality and reuse: Simulink $\rightarrow$ Modelica

From Block Diagram to Component Diagram



Component diagram in Dymola



Block diagram in Simulink

Component diagrams generalize Block diagrams
=> **The next generation of simulation tools**

Compositionality and reuse: ODE $\rightarrow$ DAE

from Simulink (ODE):
HS in state space form

$$\left\{ \begin{array}{l} x' = f(x,u) \\ y = g(x,u) \end{array} \right.$$

the state space form
depends on the context

reuse is difficult

$\longrightarrow$

to Modelica (DAE):
HS as physical balance equations

$$\left\{ \begin{array}{l} 0 = f(x',x,u) \\ 0 = g(x,u) \end{array} \right.$$

Ohm & Kirchhoff laws, bond graphs,
multi-body mechanical systems

reuse is much easier

Compositionality and reuse: ODE $\rightarrow$ DAE

- Modeling tools supporting DAE
  - Most modeling tools provide a library of predefined models ready for assembly (Mathworks/Simscape, Siemens-LMS/AmeSim, Mathematica/NDSolve)
  - Modelica comes with a full programming language that is a public standard `https://www.modelica.org/` ;
  - Simscape and NDSolve use Matlab extended with "$==$"
  - Also Spice dedicated to EDA

A sketch of Modelica and its semantics [Fritzson]

```
model SimpleDrive
   ..Rotational.Inertia   Inertia1  (J=0.002);
   ..Rotational.IdealGear IdealGear1(ratio=100)
   ..Basic.Resistor       Resistor1 (R=0.2)
      ...
equation
   connect(Inertia1.flange_b, IdealGear1.flange_a);
   connect(Resistor1.n, Inductor1.p);
      ...
end SimpleDrive;
```

```
model Resistor
   package SIunits = Modelica.SIunits;
   parameter SIunits.Resistance R = 1;
   SIunits.Voltage v;
   ..Interfaces.PositivePin p;
   ..Interfaces.NegativePin n;
equation
   0 = p.i + n.i;
   v = p.v - n.v;
   v = R*p.i;
end Resistor;
```

```
type Voltage =
     Real(quantity="Voltage",
          unit    ="V");
```

```
connector PositivePin
   package SIunits = Modelica.SIunits;
   SIunits.Voltage      v;
   flow SIunits.Current i;
end PositivePin;
```

A sketch of Modelica and its semantics [Fritzson]

- Modelica Reference v3.3:

  > *"The semantics of the Modelica language is specified by means of a set of rules for translating any class described in the Modelica language to a flat Modelica structure"*

- **the good:**
  - Semantics of continuous-time 1-mode Modelica models: Cauchy problem on the DAE resulting from the inlining of all components
  - Modelica supports multi-mode systems

```
x*x + y*y = 1;
der(x) + x + y = 0;
when x <= 0 do reinit(x,1); end;
when y <= 0 do reinit(y,x); end;
```

- **the bad:** What about the semantics of multi-mode systems?
- **and . . . :** Questionable simulations (examples later)

Examples of multi-mode systems

Cup-and-Ball game (a two-mode extension of the pendulum)

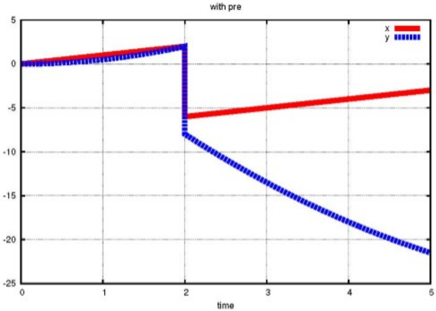
A Clutch

A Circuit Breaker

Examples of unexpected results: causal loops

A case in Modelica

```
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
```

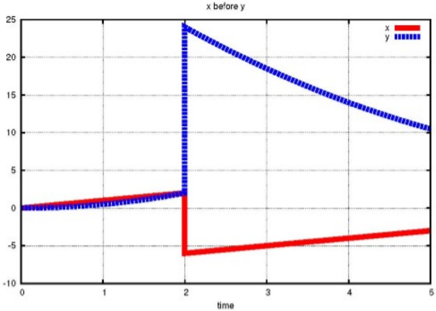


At the instant of reset, x and y each have a value defined in terms of their values just prior to the reset.

Examples of unexpected results: causal loops

A case in Modelica

```
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
```

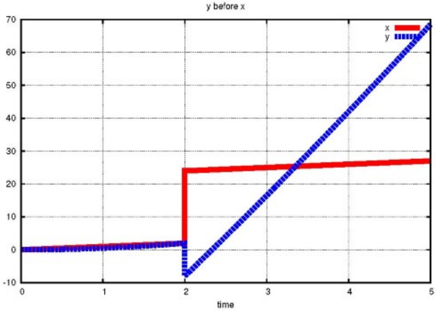


Take the pre away: At the time of reset, x and y are in cyclic dependency chain. The simulation runtime (of both OpenModelica and Dymola), chooses to reinitialize x first, with the value -6 as before, and then to reinitialize y with 24.

Examples of unexpected results: causal loops

A case in Modelica

```
model scheduling
  Real x(start=0);
  Real y(start=0);
equation
  der(x)=1;
  der(y)=x;
  when x>=2 then
    reinit(y,-4*x);
  end when;
  when x>=2 then
    reinit(x,-3*y);
  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` are each placed in their own when clause.

Examples of unexpected results: causal loops

A case in Modelica

- 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.

More strange examples later.

Examples of multi-mode systems

Cup-and-Ball game
(a two-mode
extension of
the pendulum)

$\Rightarrow$ A Clutch

A Circuit Breaker

Invoking the heritage of synchronous languages

- The constructive semantics tells how a time step should be executed
  - by scheduling atomic actions
    - evaluating expressions, forwarding control
  - according to causality constraints
    - an expression can be evaluated only if its arguments were already evaluated

Executable code follows directly

Invoking the heritage of synchronous languages

- The constructive semantics tells how a time step should be executed for multi-mode DAE systems
  - by scheduling atomic actions
    - evaluating expressions, forwarding control
    - solving algebraic systems of equations
  - according to causality constraints
    - an expression can be evaluated only if its arguments were already evaluated
    - resulting from the structural analysis

Executable code follows with some more work

The clutch example: separate analysis of each mode

$$
\left\{
\begin{array}{rlll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \\
\texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) & \text{clutch engaged} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) & \cdots \\
\texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) & \text{clutch released} \\
 & \texttt{and} & \tau_2 = 0 & (e_6) & \cdots
\end{array}
\right.
$$

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{llll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \quad \text{clutch engaged} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \quad \cdots \\
\texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \quad \text{clutch released} \\
 & \texttt{and} & \tau_2 = 0 & (e_6) \quad \cdots
\end{array}\right.$$

Mode $\gamma = \text{F}$: it is just an ODE system, nothing fancy

$$\left\{\begin{array}{ll}
\omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
\omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\tau_1 = 0 & (e_5) \\
\tau_2 = 0 & (e_6)
\end{array}\right.$$

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{llll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \quad \text{clutch engaged} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \quad \cdots \\
\texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \quad \text{clutch released} \\
 & \texttt{and} & \tau_2 = 0 & (e_6) \quad \cdots
\end{array}\right.$$

Mode $\gamma = \mathrm{T}$: it is now a DAE system

$$\left\{\begin{array}{ll}
\omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
\omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\omega_1 - \omega_2 = 0 & (e_3) \\
 & \\
\tau_1 + \tau_2 = 0 & (e_4)
\end{array}\right.$$

Looking for an execution scheme? Try a 1^{st}-order Euler scheme

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{lllll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \\
\texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) & \text{clutch engaged} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) & \cdots \\
\texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) & \text{clutch released} \\
 & \texttt{and} & \tau_2 = 0 & (e_6) & \cdots
\end{array}\right.$$

Mode $\gamma = \mathrm{T}$: it is now a dAE system

$$\left\{\begin{array}{ll}
\omega_1^\bullet = \omega_1 + \delta.f_1(\omega_1, \tau_1) & (e_1^\delta) \\
\omega_2^\bullet = \omega_2 + \delta.f_2(\omega_2, \tau_2) & (e_2^\delta) \\
\omega_1 - \omega_2 = 0 & (e_3) \\
 & \\
\tau_1 + \tau_2 = 0 & (e_4)
\end{array}\right. \tag{1}$$

Regard (1) as a transition system: for a given (ω_1, ω_2) satisfying (e_3), find $(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2)$ using eqns $(e_1^\delta, e_2^\delta, e_4)$.
We have 4 unknowns but only 3 eqns: it does not work!

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{llll} & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \quad \text{clutch engaged} \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \quad \cdots \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \quad \text{clutch released} \\ & \texttt{and} & \tau_2 = 0 & (e_6) \quad \cdots \end{array}\right.$$

Mode $\gamma = \mathrm{T}$: it is now a dAE system

$$\left\{\begin{array}{ll} \omega_1^\bullet = \omega_1 + \delta.f_1(\omega_1, \tau_1) & (e_1^\delta) \\ \omega_2^\bullet = \omega_2 + \delta.f_2(\omega_2, \tau_2) & (e_2^\delta) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \color{red}{\omega_1^\bullet = \omega_2^\bullet} & \color{red}{(e_3^\bullet)} \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array}\right. \tag{2}$$

Regard (2) as a transition system: for a given (ω_1, ω_2) satisfying (e_3),
find $(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2)$ using eqns $(e_1^\delta, e_2^\delta, e_3^\bullet, e_4)$: structurally nonsingular.
Yields a deterministic transition system;
executing it only requires an algebraic equation solver.

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{lllll} & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \\ & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) & \text{clutch engaged} \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) & \cdots \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) & \text{clutch released} \\ & \texttt{and} & \tau_2 = 0 & (e_6) & \cdots \end{array}\right.$$

Mode $\gamma = \mathrm{T}$: it is now a DAE system

$$\left\{\begin{array}{ll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' = \omega_2' & (e_3') \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array}\right. \tag{3}$$

Regard (3) as a system with dummy derivatives: for a given (ω_1, ω_2) satisfying (e_3), find $(\omega_1', \omega_2', \tau_1, \tau_2)$ using eqns (e_1, e_2, e_3', e_4): structurally nonsingular.
Yields a generalized ODE system;
executing it only requires an algebraic equation solver.

The clutch example: separate analysis of each mode

$$\left\{\begin{array}{llll} & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \quad \text{clutch engaged} \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \quad \cdots \\ \texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \quad \text{clutch released} \\ & \texttt{and} & \tau_2 = 0 & (e_6) \quad \cdots \end{array}\right.$$

Mode $\gamma = \mathrm{T}$: it is now a DAE system

$$\left\{\begin{array}{ll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' = \omega_2' & (e_3') \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array}\right. \tag{4}$$

- Adding (e_3') is called index reduction.
- It consists in finding latent equations.
- The dummy derivative approach is due to [Mattsson Söderlind 1993]

The clutch example: separate analysis of each mode

$$
\left\{
\begin{array}{llll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \quad \text{clutch engaged} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \quad \cdots \\
\texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \quad \text{clutch released} \\
 & \texttt{and} & \tau_2 = 0 & (e_6) \quad \cdots
\end{array}
\right.
$$

Mode $\gamma = \mathrm{T}$: it is now a DAE system

$$
\left\{
\begin{array}{ll}
\omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
\omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\omega_1 - \omega_2 = 0 & (e_3) \\
\omega_1' = \omega_2' & (e_3') \\
\tau_1 + \tau_2 = 0 & (e_4)
\end{array}
\right. \tag{5}
$$

- The structural analyses we performed
  - in continuous time, and
  - in discrete time using Euler schemes

  mirror each other (this is a general fact)

The clutch example: mode transitions

$$
\left\{
\begin{array}{llll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\
 & \texttt{and} & \omega_1' - \omega_2' = 0 & (e_3') \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\
\texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\
 & \texttt{and} & \tau_2 = 0 & (e_6)
\end{array}
\right.
$$

- Intuition: structural analysis in each mode is enough
- Problems:
  - reset $\neq$ initialization
    (initialization has 1 degree of freedom in mode $\gamma = \text{T}$)
  - transition *released* $\rightarrow$ *engaged* has impulsive torques
    (to adjust the rotation speeds in zero time)

The results obtained by Modelica and Mathematica are interesting

The clutch example: mode transitions

$$
\left\{
\begin{array}{rlll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\
 & \color{red}{\texttt{and}} & \color{red}{\omega_1' - \omega_2' = 0} & \color{red}{(e_3')} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\
\texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\
 & \texttt{and} & \tau_2 = 0 & (e_6)
\end{array}
\right.
$$

- Intuition: structural analysis in each mode is enough
- Problems:
  - reset $\neq$ initialization
    (initialization has 1 degree of freedom in mode $\gamma = \mathrm{T}$)
  - transition *released* $\rightarrow$ *engaged* has impulsive torques
    (to adjust the rotation speeds in zero time)

The results obtained by Modelica and Mathematica are interesting

The clutch example: mode transitions

$$
\left\{
\begin{array}{llll}
 & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\
 & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\
\texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\
 & \color{red}{\texttt{and}} & \color{red}{\omega_1' - \omega_2' = 0} & \color{red}{(e_3')} \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\
\texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\
 & \texttt{and} & \tau_2 = 0 & (e_6)
\end{array}
\right.
$$

- Intuition: structural analysis in each mode is enough
- Problems:
  - reset $\neq$ initialization
    (initialization has 1 degree of freedom in mode $\gamma = \mathrm{T}$)
  - transition *released* $\rightarrow$ *engaged* has impulsive torques
    (to adjust the rotation speeds in zero time)

The results obtained by Modelica and Mathematica are interesting

The clutch example: mode transitions

$$\left\{\begin{array}{rlll} & & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\ & \texttt{and} & \omega_1' - \omega_2' = 0 & (e_3') \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

- Intuition: structural analysis in each mode is enough
- Problems:
  - reset $\neq$ initialization
    (initialization has 1 degree of freedom in mode $\gamma = \text{T}$)
  - transition *released* $\rightarrow$ *engaged* has impulsive torques
    (to adjust the rotation speeds in zero time)

The results obtained by Modelica and Mathematica are interesting

The clutch in Modelica and Mathematica

Clutch

$$\begin{cases} & & \omega_1' = f_1(\omega_1, \tau_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \\ \texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \\ \texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 \\ & \texttt{and} & \tau_2 = 0 \end{cases}$$

Changes γ : $\mathtt{F} \rightarrow \mathtt{T} \rightarrow \mathtt{F}$ at $t = 5, 10$

When the clutch gets engaged, an impulsive torque occurs if the two rotation speeds differed before the engagement. The common speed after engagement should sit between the two speeds before it.

The clutch in Modelica and Mathematica

Clutch in Modelica

$$\begin{cases} & & \omega_1' = f_1(\omega_1, \tau_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 \\ & \texttt{and} & \tau_2 = 0 \end{cases}$$

Changes γ : F $\rightarrow$ T $\rightarrow$ 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
```

```
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;
```

The clutch in Modelica and Mathematica

Clutch in Modelica

$$\left\{\begin{array}{lll} & & \omega_1' = f_1(\omega_1, \tau_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 \\ & \texttt{and} & \tau_2 = 0 \end{array}\right.$$

Changes γ : F $\rightarrow$ T $\rightarrow$ F at $t = 5, 10$

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

```
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;
```

The clutch in Modelica and Mathematica

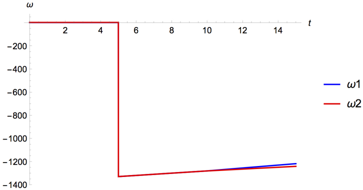
Clutch in Mathematica

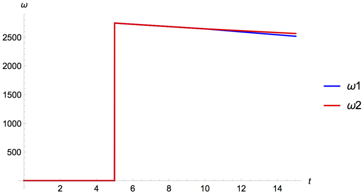
$$\left\{\begin{array}{lll} & & \omega_1' = f_1(\omega_1, \tau_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 \\ & \texttt{and} & \tau_2 = 0 \end{array}\right.$$

Changes γ : F $\rightarrow$ T $\rightarrow$ F at $t = 5, 10$

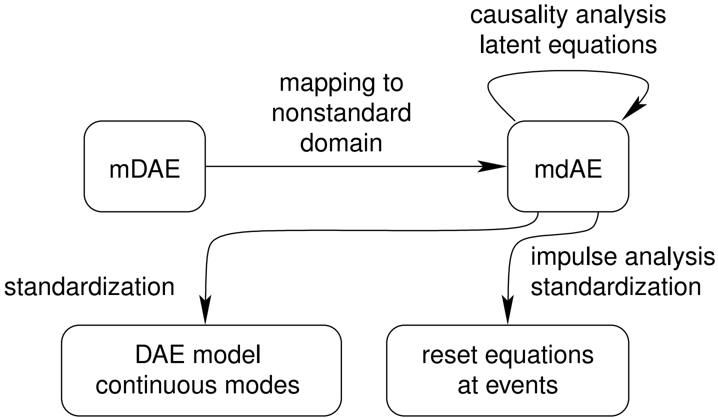
The simulation does not crash but yields meaningless results highly sensitive to little variations of some parameters.
Suggests that a cold restart, not a reset, is performed.

```
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] == 1.5, s[0] == 0,
   WhenEvent[t == 5,
         s[t] -> 1
         ] },
   w1, w2, t1, t2,s,
   t, 0, 7, DiscreteVariables -> s]
```

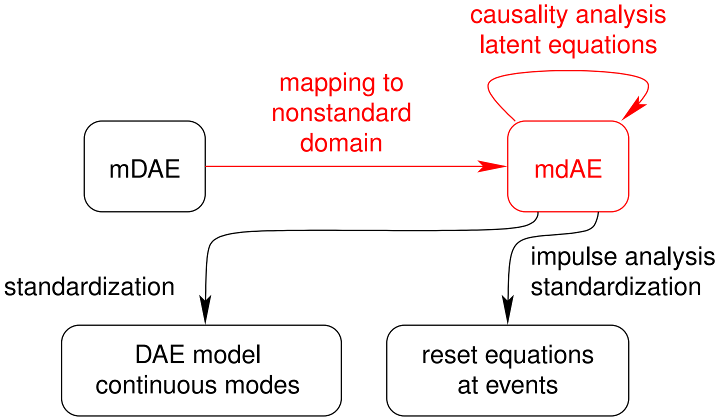




Overview of our approach



Nonstandard structural analysis



Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\mathrm{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

$$\left\{\begin{array}{rlll} & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1,\tau_1) & (e_1^\partial) \\ & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2,\tau_2) & (e_2^\partial) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\text{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

$$\left\{\begin{array}{lll l}
 & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1,\tau_1) & (e_1^\partial) \\
 & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2,\tau_2) & (e_2^\partial) \\
\texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\
\texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\
 & \texttt{and} & \tau_2 = 0 & (e_6)
\end{array}\right.$$

- If $\gamma = \mathtt{F}$ then we have an ODE system: easy
- If $\gamma = \mathtt{T}$, two cases occur, depending on whether (e_3) is satisfied or not, by the states ω_1, ω_2

Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\text{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

$$\left\{\begin{array}{llll} & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1,\tau_1) & (e_1^\partial) \\ & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2,\tau_2) & (e_2^\partial) \\ \texttt{when}\ \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

Case (e_3) is satisfied by the states ω_1, ω_2

- block $\{(e_1^\partial), (e_2^\partial), (e_4)\}$ has 4 unknowns $\omega_i^\bullet, \tau_i$
- need to find latent equations: add

$$\texttt{when}\ \gamma\ \texttt{do}\ \omega_1^\bullet - \omega_2^\bullet = 0 \qquad (e_3^\bullet)$$

  and we conclude as for the engaged mode: use block $\{(e_1^\partial), (e_2^\partial), (e_3^\bullet), (e_4)\}$ to evaluated the 4 unknowns $\omega_i^\bullet, \tau_i$

Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\text{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

$$\left\{\begin{array}{rlll} & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1, \tau_1) & (e_1^\partial) \\ & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2, \tau_2) & (e_2^\partial) \\ \texttt{when } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 & (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

Case (e_3) is not satisfied by the states ω_1, ω_2

- (e_3) is an overconstrained system
- Causality Principle:
  A guard must be evaluated before the equation it controls
- Applying the causality principle leads to
  Shifting forward the body of (e_3)

Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\text{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

$$\left\{\begin{array}{lll|l} & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1,\tau_1) & (e_1^\partial) \\ & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2,\tau_2) & (e_2^\partial) \\ \texttt{when}\ \gamma & \texttt{do} & \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

Case (e_3) is not satisfied by the states ω_1, ω_2

- (e_3) is an overconstrained system
- Causality Principle:
  A guard must be evaluated before the equation it controls
- Applying the causality principle leads to
  Shifting forward the body of (e_3)
- We conclude as before

Nonstandard structural analysis

∂ infinitesimal; ${}^\star\mathbb{T} =_{\text{def}} \{n.\partial \mid n \in {}^\star\mathbb{N}\}$; nonstandard clutch model:

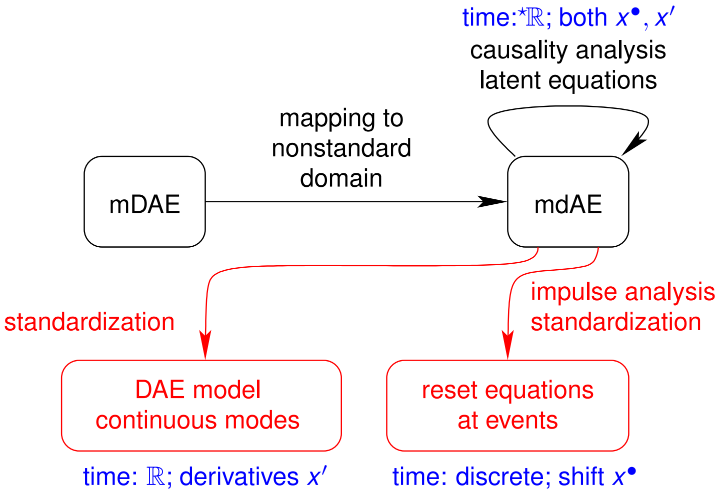
$$\left\{\begin{array}{llll}
 & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1,\tau_1) & (e_1^\partial) \\
 & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2,\tau_2) & (e_2^\partial) \\
\texttt{when}\ \gamma & \texttt{do} & \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\
 & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\
\texttt{when not}\ \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\
 & \texttt{and} & \tau_2 = 0 & (e_6)
\end{array}\right.$$

Execution Scheme 6 for Nonstandard model: ensures $\omega_1 = \omega_2$.

Require: ω_1 and ω_2.

1: **if** γ **then**
2: $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet) \leftarrow$ Solve $\{e_1^\partial, e_2^\partial, e_3^\bullet, e_4\}$
3: **else**
4: $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet) \leftarrow$ Solve $\{e_1^\partial, e_2^\partial, e_5, e_6\}$
5: **end if**
6: Tick $\triangleright$ Move to next step

Back-Standardization



Back-Standardization

We start from the nonstandard clutch model:

$$\left\{\begin{array}{rlll} & & \omega_1^\bullet = \omega_1 + \partial.f_1(\omega_1, \tau_1) & (e_1^\partial) \\ & & \omega_2^\bullet = \omega_2 + \partial.f_2(\omega_2, \tau_2) & (e_2^\partial) \\ \texttt{when } \gamma & \texttt{do} & (\omega_1 - \omega_2 = 0) & ((e_3)) \\ & \texttt{and} & \color{red}{\omega_1^\bullet - \omega_2^\bullet = 0} & \color{red}{(e_3^\bullet)} \\ & \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{when not } \gamma & \texttt{do} & \tau_1 = 0 & (e_5) \\ & \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right.$$

Back-Standardization

Within continuous modes:

- time is $\mathbb{R}$
- nonstandard derivatives $\rightarrow$ standard derivatives: $e_i^\partial \rightarrow e_i, i = 1, 2$ (easy)
- what about $e_3^\bullet : \omega_1^\bullet = \omega_2^\bullet$?

$$\begin{array}{rrcl} \omega_1^\bullet = \omega_2^\bullet \text{ expands as:} & \omega_1 + \partial.\omega_1' & = & \omega_2 + \partial.\omega_2' \\ \text{from previous step:} & \omega_1 & = & \omega_2 \\ \text{which implies, by subtracting:} & \omega_1' & = & \omega_2' \end{array}$$

- we thus recover the dynamics for the engaged mode, as obtained by the dummy derivatives method:

$$\left\{ \begin{array}{ll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' = \omega_2' & (e_3') \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array} \right.$$

Back-Standardization

At events:

- Time is discrete: $t, t^\bullet, t^{\bullet 2}, \ldots$; all the $t^{\bullet k}$ occur at time t
- Equation $e_3^\bullet : \omega_1^\bullet = \omega_2^\bullet$ makes no trouble
- This time the problem is with the $(e_1^\partial, e_2^\partial)$, due to the ∂ in space

$$\left\{ \begin{array}{ll} \omega_1^\bullet = \omega_1 + \partial . f_1(\omega_1, \tau_1) & (e_1^\partial) \\ \omega_2^\bullet = \omega_2 + \partial . f_2(\omega_2, \tau_2) & (e_2^\partial) \end{array} \right. \tag{6}$$

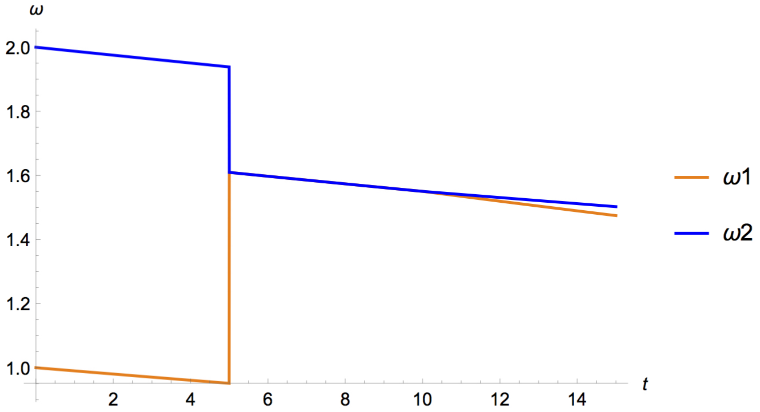
  We must eliminate ∂ from (6).

- We have developed a systematic approach using Taylor expansions for the f_i. For the simple case where $f_i(\omega_i, \tau_i) = a_i\omega_i + b_i\tau_i$, we get

$$\begin{aligned} \omega_i^\bullet &= \frac{b_2\omega_1 + b_1\omega_2}{b_1 + b_2} + \partial . \frac{a_1 b_2\omega_1 + a_2 b_1\omega_2}{b_1 + b_2} \\ st(\omega_i^\bullet) &= \frac{b_2\omega_1 + b_1\omega_2}{b_1 + b_2} \end{aligned}$$

  and the torques are impulsive, of order $0(\partial^{-1})$

Our simulation results



mode changes $\gamma : \mathsf{F} \rightarrow \mathsf{T} \rightarrow \mathsf{F}$ at $t = 5, 10$

mDAE, mdAE: nonstandard denotational semantics

$$
\begin{aligned}
X^{(\Sigma)} \quad &=_{\text{def}} \quad \bigcup_{m\in\Sigma} X^{(m)},\ \text{e.g., for } x \in X : x^{(\bullet\bullet'\bullet'')} \\
\{X\} \quad &=_{\text{def}} \quad X^{(\{\bullet,'\}^*)}\ ,\ \text{where } m \in \{\bullet,'\}^*
\end{aligned}
$$

Definition mDAE:

$s ::= e \mid s_1 , s_2$ where $e ::= \texttt{if}\ \gamma\ \texttt{do}\ f{=}0$, X finite set of variables, and

- f is a scalar smooth function over $\{X\}$;
- γ is a predicate over $\{X\}$;
- s_1 , s_2 denotes the conjunction of s_1 and s_2.

A mode, in an mDAE, is a valuation of its guards.

For a guarded equation e, $f{=}0$ (resp. γ) is denoted by e_f (resp. e_γ).

$$
\text{nonstandard mdAE} \quad =_{\text{def}} \quad \text{mDAE}\left[x' \mapsto \frac{x^\bullet - x}{\partial}\right] \quad .
$$

Since an mdAE is a transition system, we know what its denotational semantics is

mdAE: nonstandard constructive semantics

The constructive semantics tells how a time step should be effectively performed by scheduling atomic actions according to causality constraints.

Abstract Scott domain: $\mathcal{D} = \{\perp, \mathrm{F}, \mathrm{T}\}$ with $\perp < \mathrm{F}, \mathrm{T}$, where:

- for variables: $\perp \equiv$"not evaluated", $\mathrm{T} \equiv$"evaluated"
- for guards: $\perp \equiv$"not evaluated", $\mathrm{T}/\mathrm{F} \equiv$"evaluated"
- for g_eqns: $\perp \equiv$"not evaluated", $\mathrm{T} \equiv$"solved", $\mathrm{F} \equiv$"dead" (because $\gamma = \mathrm{F}$)

Atomic actions consist of:

- evaluating guards
- ⇒ solving blocks of equations
- ⇒ massaging equations (shifting, finding latent equations in dAE systems)
- performing a tick

mdAE: nonstandard constructive semantics

The constructive semantics tells how a time step should be effectively performed by scheduling atomic actions according to causality constraints.

Abstract Scott domain: $\mathcal{D} = \{\perp, \mathtt{F}, \mathtt{T}\}$ with $\perp \; < \; \mathtt{F}, \mathtt{T}$, where:

- for variables: $\perp$ $\equiv$“not evaluated”, $\mathtt{T}$ $\equiv$“evaluated”
- for guards: $\perp$ $\equiv$“not evaluated”, $\mathtt{T}/\mathtt{F}$ $\equiv$“evaluated”
- for g_eqns: $\perp$ $\equiv$“not evaluated”, $\mathtt{T}$ $\equiv$“solved”, $\mathtt{F}$ $\equiv$“dead” (because $\gamma = \mathtt{F}$)

Atomic actions consist of:

- evaluating guards

$\Rightarrow$ solving blocks of equations

$\Rightarrow$ massaging equations (shifting, finding latent equations in dAE systems)

- performing a tick

mdAE: nonstandard constructive semantics

The constructive semantics tells how a time step should be effectively performed by scheduling atomic actions according to causality constraints.

Abstract Scott domain: $\mathcal{D} = \{\bot, \mathtt{F}, \mathtt{T}\}$ with $\bot < \mathtt{F}, \mathtt{T}$, where:

- for variables: $\bot$ ≡"not evaluated", $\mathtt{T}$ ≡"evaluated"
- for guards: $\bot$ ≡"not evaluated", $\mathtt{T}/\mathtt{F}$ ≡"evaluated"
- for g_eqns: $\bot$ ≡"not evaluated", $\mathtt{T}$ ≡"solved", $\mathtt{F}$ ≡"dead" (because $\gamma = \mathtt{F}$)

Atomic actions consist of:

- evaluating guards

⇒ solving blocks of equations

⇒ massaging equations (shifting, finding latent equations in dAE systems)

- performing a tick

mdAE: nonstandard constructive semantics

- Status: $\sigma : x/\gamma/e \mapsto \mathcal{D}$ satisfying coherence conditions (causality):

$$\sigma(\gamma(x_1,\ldots,x_n)) = \bot \text{ if } \exists i, \sigma(x_i) = \bot$$

$$\sigma(\texttt{if } \gamma \texttt{ do } f{=}0) \begin{cases} = \bot & \text{if } \sigma(\gamma) = \bot \\ = \mathsf{F} & \text{if } \sigma(\gamma) = \mathsf{F} \\ \in \{\bot, \sigma(f{=}0)\} & \text{if } \sigma(\gamma) = \mathsf{T} \end{cases}$$

  where $\sigma(f{=}0)$ is a shorthand for

$$\begin{cases} \bot & \text{if } \exists i.\sigma(x_i) = \bot \\ \mathsf{T} & \text{otherwise} \end{cases},$$

  and the x_i are the arguments of f.

- Constructive semantics: $\sigma_0 < \sigma_1 < \cdots < \sigma_k < \sigma_{k+1} < \cdots < \sigma_K$
- Success:
  - no g_eqn remains $\bot$ in $\sigma_K \Rightarrow$ the mode is known $\Rightarrow$ we know what the leading variables are;
  - no leading variable remains $\bot$ in σ_K

mdAE: nonstandard constructive semantics

- Status: $\sigma : x/\gamma/e \mapsto \mathcal{D}$ satisfying coherence conditions (causality):

$$\sigma(\gamma(x_1,\ldots,x_n)) = \perp \text{ if } \exists i, \sigma(x_i) = \perp$$

$$\sigma(\texttt{if } \gamma \texttt{ do } f{=}0) \begin{cases} = \perp & \text{if } \sigma(\gamma) = \perp \\ = \mathrm{F} & \text{if } \sigma(\gamma) = \mathrm{F} \\ \in \{\perp, \sigma(f{=}0)\} & \text{if } \sigma(\gamma) = \mathrm{T} \end{cases}$$

  where $\sigma(f{=}0)$ is a shorthand for

$$\begin{cases} \perp & \text{if } \exists i.\sigma(x_i) = \perp \\ \mathrm{T} & \text{otherwise} \end{cases},$$

  and the x_i are the arguments of f.

- Constructive semantics: $\sigma_0 < \sigma_1 < \cdots < \sigma_k < \sigma_{k+1} < \cdots < \sigma_K$

- Success:
  - no g_eqn remains $\perp$ in $\sigma_K \Rightarrow$ the mode is known $\Rightarrow$ we know what the leading variables are;
  - no leading variable remains $\perp$ in σ_K

mdAE: nonstandard constructive semantics

- Status: $\sigma : x/\gamma/e \mapsto \mathcal{D}$ satisfying coherence conditions (causality):

$$\sigma(\gamma(x_1,\ldots,x_n)) = \perp \text{ if } \exists i, \sigma(x_i) = \perp$$

$$\sigma(\texttt{if } \gamma \texttt{ do } f{=}0) \begin{cases} = \perp & \text{if } \sigma(\gamma) = \perp \\ = \mathsf{F} & \text{if } \sigma(\gamma) = \mathsf{F} \\ \in \{\perp, \sigma(f{=}0)\} & \text{if } \sigma(\gamma) = \mathsf{T} \end{cases}$$

  where $\sigma(f{=}0)$ is a shorthand for

$$\begin{cases} \perp & \text{if } \exists i. \sigma(x_i) = \perp \\ \mathsf{T} & \text{otherwise} \end{cases},$$

  and the x_i are the arguments of f.

- Constructive semantics: $\sigma_0 < \sigma_1 < \cdots < \sigma_k < \sigma_{k+1} < \cdots < \sigma_K$
- Success:
  - no g_eqn remains $\perp$ in σ_K $\Rightarrow$ the mode is known $\Rightarrow$ we know what the leading variables are;
  - no leading variable remains $\perp$ in σ_K

mdAE: nonstandard constructive semantics

Algorithm 7 Building Constructive Semantics

Require: mdAE S and an initial status σ and context Δ

$V \leftarrow \textbf{ScottVars}[S]$
$V_\perp \leftarrow \{v \in V \mid \sigma(v) = \perp\}$ ▷ Scott vars. for eval.
while $V_\perp \neq \emptyset$ **do**
  $\forall \gamma \in V_\perp$. s.t. $\sigma(\gamma) = \perp$, $\textbf{Eval}[\gamma, \sigma]$ ▷ nondet. eval
  if $\forall \gamma \in V_\perp . \sigma(\gamma) \neq \perp$ **then** ▷ mode known
    $V_\perp \leftarrow V_\perp \setminus (\textbf{Ld}[\sigma])^c$ ▷ discard irrelevant vars.
  end if
  $\sigma \leftarrow \pi_\Delta(\sigma)$ ▷ project over Δ
  $F \leftarrow \{e_f \mid \sigma(e) = \perp \wedge \sigma(e_\gamma) = \top\}$ ▷ select active eqns.
  $\{B_\mathfrak{e}, B_\mathfrak{o}, B_\mathfrak{u}\} \leftarrow \textbf{BLT}[F, \sigma]$ ▷ BLT decomposition
  if $\exists b \in B_\mathfrak{e}$ **then** ▷ solving blocks
    $\forall y \in \text{Vars}[b]$, $\sigma(y) \leftarrow \top$ ▷ update σ
    $\forall e \in \text{Eq}[b]$, $\sigma(e) \leftarrow \top$ ▷ update σ
    $V_\perp \leftarrow V_\perp \setminus (\text{Vars}[b] \cup \text{Eq}[b])$ ▷ update $V_\perp$
  else if $\exists b \in B_\mathfrak{o}$ **then** ▷ overdet. subsystems
    $(F, V_\perp) \leftarrow (F, V_\perp)[e_b \mapsto e_b^\bullet]_{\forall b \in B_\mathfrak{o}}$ ▷ fward. shift
  else if $\exists b \in B_\mathfrak{u}$ **then** ▷ underdet. subsystems
    $F \leftarrow F \cup \textbf{LatentEq}[b]$ ▷ add latent eq.
  end if
end while
Tick

The constructive semantics: sketch

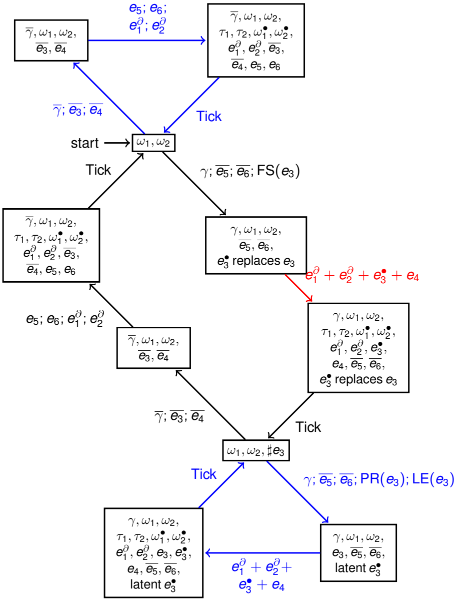
For S a multi-mode DAE system

- map $S \mapsto S^{\partial}$ through the substitution $x' \mapsto \frac{1}{\partial}(x^{\bullet} - x)$
- build the constructive semantics:
  1. for each possible initial status (value for every state) and context (equations that were proved satisfied at previous steps)
  2. evaluate enabled guards ($\in \{\mathtt{F}, \mathtt{T}\}$) and keep/discard active/dead equations and clean the context
  3. when all guards evaluated, the mode is known
  4. perform Block Triangular Form (BTF) structural analysis
  5. if exists a regular block, solve it and return to 2.
  6. if exists an overconstrained block shift equations and return to 4.
  7. if exists an underconstrained block look for latent equations, add them and return to 4.
  8. Tick: update next initial status and context
- perform back-standardization (not easy)

Theoretical results (to be done)

1. **Soundness w.r.t. nonstandard semantics**: future work.
    - Proving that our algorithm actually executes the nonstandard denotational semantics
    - There are subtleties, due to the shifting of overconstrained equations
2. **Soundness w.r.t. standard semantics**: preliminary results
    - No reference denotational semantics exists for mDAE systems
    - Hence there is nothing to compare with
    - So far the best we can expect is to prove that we actually execute the right dynamics in each continuous mode. There is nothing we can say about events and resets.

Clutch: nonstandard constructive semantics



Clutch: (standard) executable code

start $\longrightarrow$

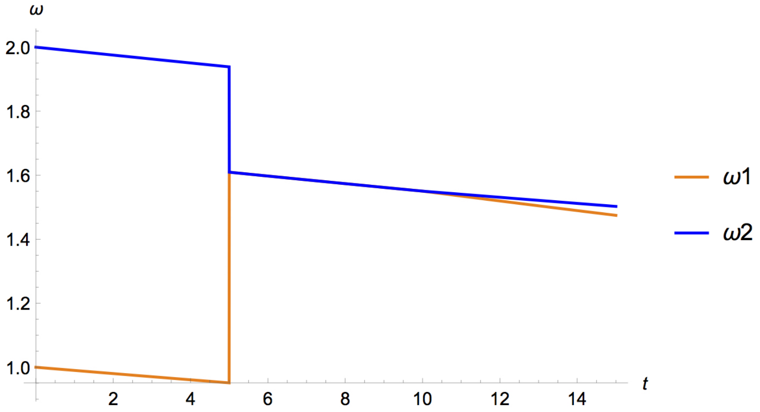
mode $\neg\gamma$: index 0
$\tau_1 = 0;\ \tau_2 = 0;$
$\omega_1' = a_1\omega_1 + b_1\tau_1;$
$\omega_2' = a_2\omega_2 + b_2\tau_2$

when $\neg\gamma$ **do**
$\tau_1 = 0;\ \tau_2 = 0;$
$\omega_1 = \omega_1^-;$
$\omega_2 = \omega_2^-$
done

when γ **do**
$\tau_1 = \text{NaN};\ \tau_2 = \text{NaN};$
$\omega_1 = \frac{b_2\omega_1^- + b_1\omega_2^-}{b_1+b_2};$
$\omega_2 = \omega_1$
done

mode γ : index 1
$\tau_1 = (a_2\omega_2 - a_1\omega_1)/(b_1 + b_2); \tau_2 = -\tau_1;$
$\omega_1' = a_1\omega_1 + b_1\tau_1;\ \omega_2' = a_2\omega_2 + b_2\tau_2;$
constraint $\omega_1 - \omega_2 = 0$

Clutch: (standard) executable code



Conclusions (about the “Modelica” family)

- Modelica is much more powerful than classical (Simulink-like) modeling:
  - models for simulation by assembling sub-models from libraries
  - DAEs, multi-mode

- The compilation of Modelica with its multi-mode extension is difficult
  - problems in Modelica tools
  - we proposed a systematic approach (more to be done)

- Other uses of Modelica
  - **Requirements**: expressing abstract properties of systems as an early phase of system design. Requires supporting under-determined multi-mode DAE systems (less equations than variables)
  - **Fault detection and diagnosis**: generating parity models $F(X$ and derivatives, $Y, U)$ where some of the Y's and U's are observed; check if $F = 0$ holds when feeding with measurements.

  Requires extensions of Modelica compilation techniques.

Conclusions (about the "Modelica" family)

- Modelica is much more powerful than classical (Simulink-like) modeling:
  - models for simulation by assembling sub-models from libraries
  - DAEs, multi-mode
- The compilation of Modelica with its multi-mode extension is difficult
  - problems in Modelica tools
  - we proposed a systematic approach (more to be done)
- Other uses of Modelica
  - **Requirements**: expressing abstract properties of systems as an early phase of system design. Requires supporting under-determined multi-mode DAE systems (less equations than variables)
  - **Fault detection and diagnosis**: generating parity models $F(X$ and derivatives, $Y, U)$ where some of the Y's and U's are observed; check if $F = 0$ holds when feeding with measurements.

  Requires extensions of Modelica compilation techniques.

Conclusions (about the "Modelica" family)

- Modelica is much more powerful than classical (Simulink-like) modeling:
  - models for simulation by assembling sub-models from libraries
  - DAEs, multi-mode
- The compilation of Modelica with its multi-mode extension is difficult
  - problems in Modelica tools
  - we proposed a systematic approach (more to be done)
- Other uses of Modelica
  - **Requirements**: expressing abstract properties of systems as an early phase of system design. Requires supporting under-determined multi-mode DAE systems (less equations than variables)
  - **Fault detection and diagnosis**: generating parity models $F(X$ and derivatives, $Y, U)$ where some of the Y's and U's are observed; check if $F = 0$ holds when feeding with measurements.

  Requires extensions of Modelica compilation techniques.

Conclusions (about the "Modelica" family)

- Modelica is much more powerful than classical (Simulink-like) modeling:
  - models for simulation by assembling sub-models from libraries
  - DAEs, multi-mode
- The compilation of Modelica with its multi-mode extension is difficult
  - problems in Modelica tools
  - we proposed a systematic approach (more to be done)
- Other uses of Modelica
  - **Requirements**: expressing abstract properties of systems as an early phase of system design. Requires supporting under-determined multi-mode DAE systems (less equations than variables)
  - **Fault detection and diagnosis**: generating parity models F(X and derivatives, Y, U) where some of the Y's and U's are observed; check if $F = 0$ holds when feeding with measurements.

Requires extensions of Modelica compilation techniques.

Conclusions (about the "Modelica" family)

- Modelica is much more powerful than classical (Simulink-like) modeling:
  - models for simulation by assembling sub-models from libraries
  - DAEs, multi-mode
- The compilation of Modelica with its multi-mode extension is difficult
  - problems in Modelica tools
  - we proposed a systematic approach (more to be done)
- Other uses of Modelica
  - **Requirements**: expressing abstract properties of systems as an early phase of system design. Requires supporting under-determined multi-mode DAE systems (less equations than variables)
  - **Fault detection and diagnosis**: generating parity models $F(X$ and derivatives, $Y, U)$ where some of the Y's and U's are observed; check if $F = 0$ holds when feeding with measurements.

Requires extensions of Modelica compilation techniques.

Ite Missa Est

Deo Gratias

Ite Missa Est

Deo Gratias