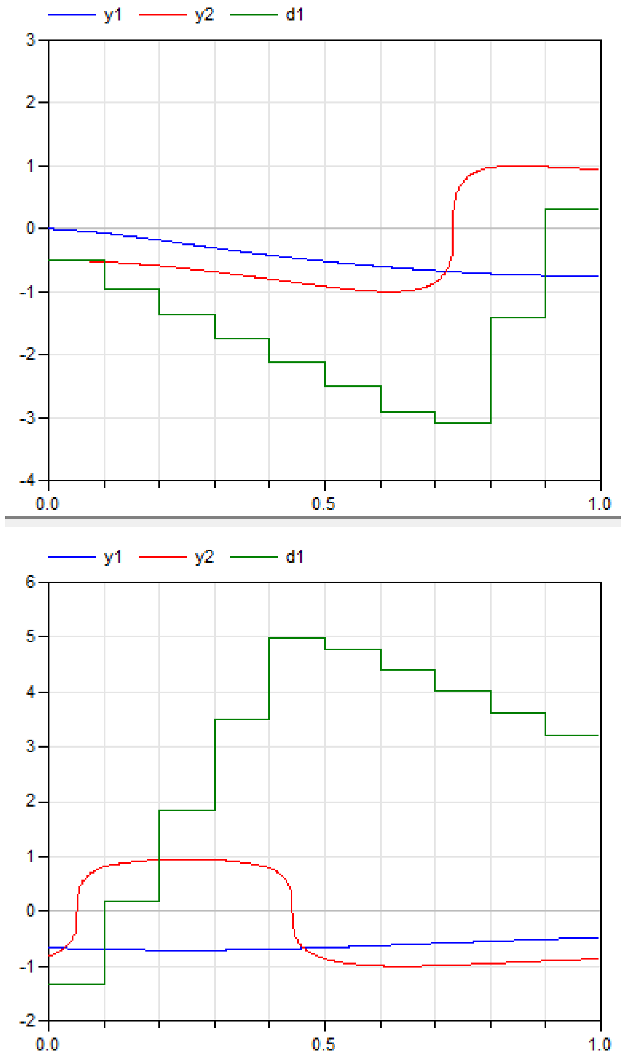
Initialization of Equation-Based Hybrid Models within OpenModelica

Lennart Ochel Bernhard Bachmann
Bielefeld University of Applied Sciences

Outline

- Modelica and Initialization
- Mathematical Representation
- Methods within OpenModelica
- Conclusion and Outlook



Modelica and Initialization

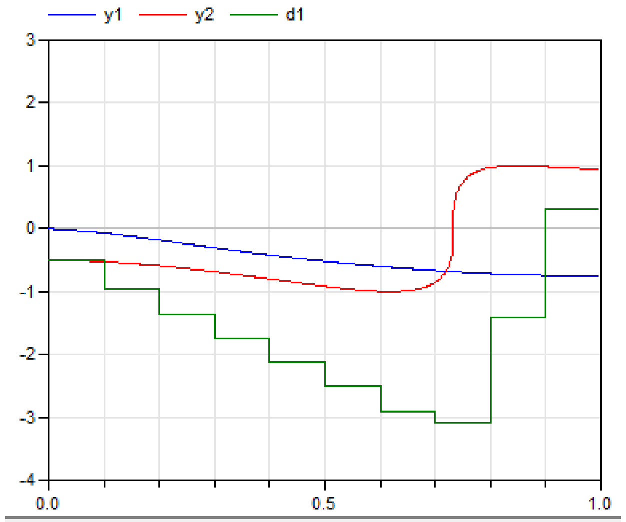
Modelica and Initialization

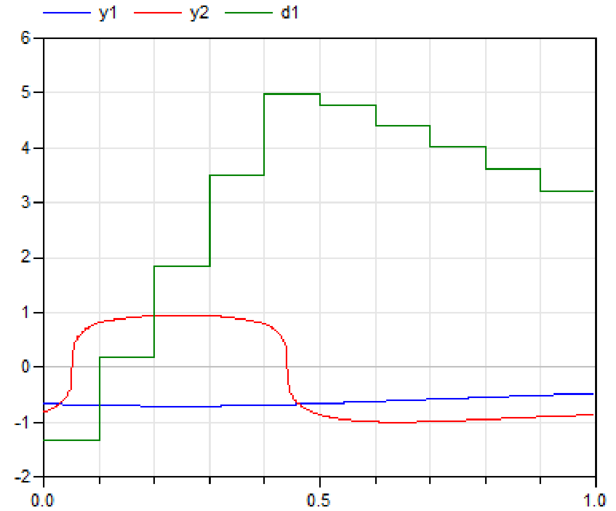
Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```





Modelica and Initialization

Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```

Continous Part

- system of differential algebraic equations
- **initial value problem** needs to be solved

e.g. Euler method

$$x_{n+1} = x_{\mathrm{n}} + (t_{n+1} - t_n) \cdot \dot{x}_n$$

Modelica and Initialization

Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```

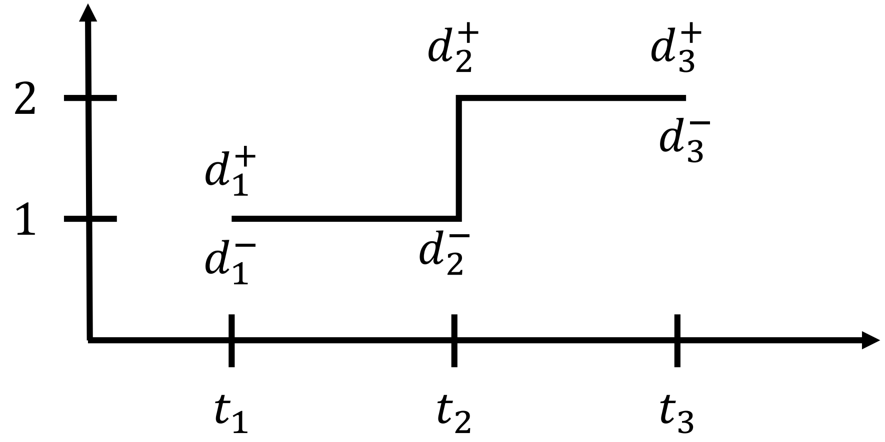
Discrete Part

- **left limit** is assumed to be known

$$d_n^+ = \xi(d_n^-, t_n, \dots)$$

e.g.: $d_2^+ = d_2^- + 1$

$d_3^+ = d_3^-$



Modelica and Initialization

Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```

```
  d1 = pre(d1) - y1 + y2; // active
or
  d1 = pre(d1);           // inactive
```

Discrete Part

- when-clauses are only active during initialization, if they are explicitly enabled using the `initial()` operator

Modelica and Initialization

Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```

Variable Attributes

- initial equations can be implicitly declared using the fixed attribute
- initial guesses can be provided using the start attribute

v(start=v^{start})		fixed=true	fixed=false
type of v	continuous	initial equation: $v = v^{start}$	initial guess of v
	discrete	initial equation: $pre(v) = v^{start}$	initial guess of $pre(v)$

Modelica and Initialization

Example

```
model MathRep
  Real x1(start=2.0, fixed=true),
       x2(start=4);
  Real y1, y2, y3(start=-1.5);
  Real d1;

initial equation
  pre(d1) = -0.5 + y1;

equation
  0 = -y2 + sin(y3);
  der(x1) = sqrt(x1) + time - d1;
  0 = x1 + y2 + y3 + 1;
  0 = x1 + y1 + x1*y1;
  when {initial(), sample(0.1, 0.1)} then
    d1 = pre(d1) - y1 + y2;
  end when;
  der(x2) = x1 + y1;
end MathRep;
```

Initial Equation/Algorithm

- additional contraints that are just used for initialization
- when-clauses are not allowed
- pure algebraic system

Mathematical Representation

Mathematical Representation

Variables

name	description
$\underline{x}(t), \underline{\dot{x}}(t)$	vector of all states/derived states
$\underline{y}(t)$	vector of all algebraic variables
$\underline{d}(t), \underline{d}^{pre}(t)$	vector of all discrete variables
$\underline{p} = \left(\underline{p}^{fixed} \quad \underline{p}^{free}\right)^{\top}$	vector of all parameters
t	simulation time
t_0	initialization time

$$\underline{\omega}(t_0) := \left(\underline{x}(t_0) \quad \underline{p}^{free} \quad \underline{d}^{pre}(t_0)\right)^{\top}$$
$$\underline{z}(t) := \left(\underline{\dot{x}}(t) \quad \underline{y}(t) \quad \underline{d}(t)\right)^{\top}$$

Mathematical Representation

Variables

name	description
$\underline{x}(t), \underline{\dot{x}}(t)$	vector of all states/derived states
$\underline{y}(t)$	vector of all algebraic variables
$\underline{d}(t), \underline{d}^{pre}(t)$	vector of all discrete variables
$\underline{p} = \left(\underline{p}^{fixed} \quad \underline{p}^{free}\right)^\top$	vector of all parameters
t	simulation time
t_0	initialization time

Equation Systems

$$\underline{0} \overset{!}{=} \underline{f}\left(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{d}(t), \underline{d}^{pre}(t), \underline{p}, t\right)$$

$$\underline{z}(t) = \underline{g}\left(\underline{x}(t), \underline{d}^{pre}(t), \underline{p}, t\right)$$
$$\Leftrightarrow \underline{z}(t) = \underline{g}\left(\underline{\omega}(t), \underline{p}^{fixed}, t\right)$$

$$\underline{0} \overset{!}{=} \underline{h}^{res} := \underline{h}\left(\underline{x}(t_0), \underline{\dot{x}}(t_0), \underline{y}(t_0), \underline{d}(t_0), \underline{d}^{pre}(t_0), \underline{p}, t_0\right)$$
$$\Leftrightarrow \underline{0} \overset{!}{=} \underline{h}^{res} := \underline{h}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)$$

$$\underline{\omega}(t_0) := \left(\underline{x}(t_0) \quad \underline{p}^{free} \quad \underline{d}^{pre}(t_0)\right)^\top$$
$$\underline{z}(t) := \left(\underline{\dot{x}}(t) \quad \underline{y}(t) \quad \underline{d}(t)\right)^\top$$

Mathematical Representation

Numeric Approach

$$\min_{\underline{\omega}} \phi\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0\right)^2 \rightarrow 0$$

s.t.

$$\underline{z}(t_0) = \underline{g}(\underline{\omega}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0)$$
$$\underline{\omega}^{min} \leq \underline{\omega} \leq \underline{\omega}^{max}$$

$\dim\left(\underline{\omega}(t_0)\right)$ must be less than or equal to the number of initial equations

$$\underline{\omega}(t_0) \coloneqq \begin{pmatrix}\underline{x}(t_0) & \underline{p}^{free} & \underline{d}^{pre}(t_0)\end{pmatrix}^{\top}$$
$$\underline{z}(t) \coloneqq \begin{pmatrix}\underline{\dot{x}}(t) & \underline{y}(t) & \underline{d}(t)\end{pmatrix}^{\top}$$

Mathematical Representation

Numeric Approach

$$\min_{\underline{\omega}} \phi\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0\right)^2 \to 0$$

s.t.

$$\underline{z}(t_0) = \underline{g}(\underline{\omega}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0)$$
$$\underline{\omega}^{min} \leq \underline{\omega} \leq \underline{\omega}^{max}$$

$\dim\left(\underline{\omega}(t_0)\right)$ must be less than or equal to the number of initial equations

Symbolic Approach

$$\begin{pmatrix} \underline{z}(t_0) \\ \underline{0} \end{pmatrix} = \begin{pmatrix} \underline{g}\left(\underline{\omega}(t_0), \underline{d}(t), \underline{p}^{fixed}, t\right) \\ \underline{h}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0\right) \end{pmatrix}$$

solving for $\underline{z}(t_0)$ and $\underline{\omega}(t_0)$

$\dim\left(\underline{\omega}(t_0)\right)$ has to be equal to the number of initial equations

$$\underline{\omega}(t_0) \coloneqq \begin{pmatrix} \underline{x}(t_0) & \underline{p}^{free} & \underline{d}^{pre}(t_0) \end{pmatrix}^\top$$
$$\underline{z}(t) \coloneqq \begin{pmatrix} \underline{\dot{x}}(t) & \underline{y}(t) & \underline{d}(t) \end{pmatrix}^\top$$

Numeric Method

Numerical Method

Basic Approach

$$\min_{\underline{\omega}(t_0)} \phi\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \to 0$$

s.t.

$$\underline{z}(t_0) = \underline{g}\left(\underline{\omega}(t_0), \underline{p}^{fixed}, t_0\right)$$
$$\underline{\omega}^{min} \leq \underline{\omega}(t_0) \leq \underline{\omega}^{max}$$

with

$$\phi(.) = \sum_i h_i^{res}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)^2$$

Numerical Method

Basic Approach

$$\min_{\underline{\omega}(t_0)} \phi\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \rightarrow 0$$

s.t.

$$\underline{z}(t_0) = \underline{g}\left(\underline{\omega}(t_0), \underline{p}^{fixed}, t_0\right)$$
$$\underline{\omega}^{min} \leq \underline{\omega}(t_0) \leq \underline{\omega}^{max}$$

with

$$\phi(.) = \sum_i h_i^{res}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)^2$$

Start Value Homotopy Approach

$$\phi(.) = (1-\lambda)\cdot\phi_0 + \lambda\cdot\phi_1$$
$$\lambda \in [0;1] \subset R$$

with

$$\phi_0(.) = \sum_{\forall v}(v - v^{start})^2$$
$$\phi_1(.) = \sum_i h_i^{res}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)^2$$

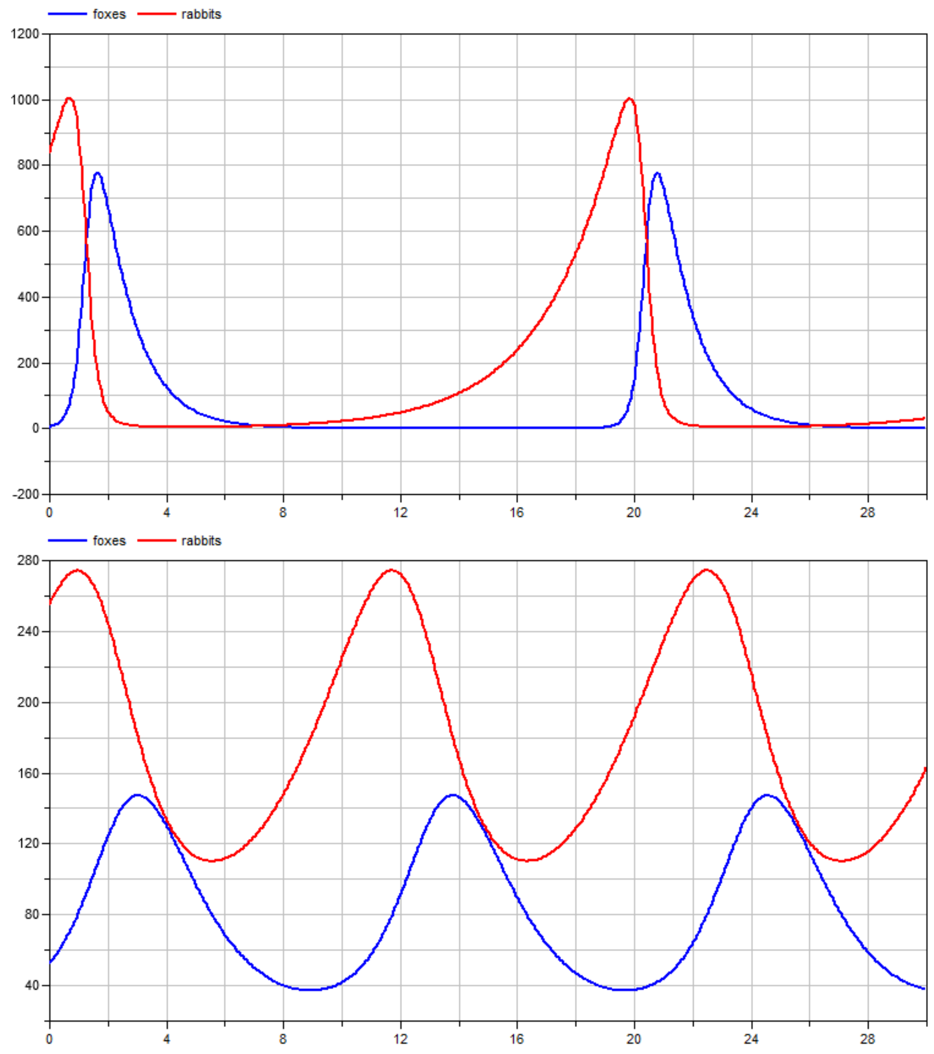
Numerical Method

Example

```
model forest
  Real foxes;
  Real rabbits;
  Real population;
  Real value;
  […]

initial equation
  der(foxes) = 20;
  value = 11000;

equation
  der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
  der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
  population   = foxes+rabbits;
  value        = priceFox*foxes + priceRabbit*rabbits;
end forest;
```



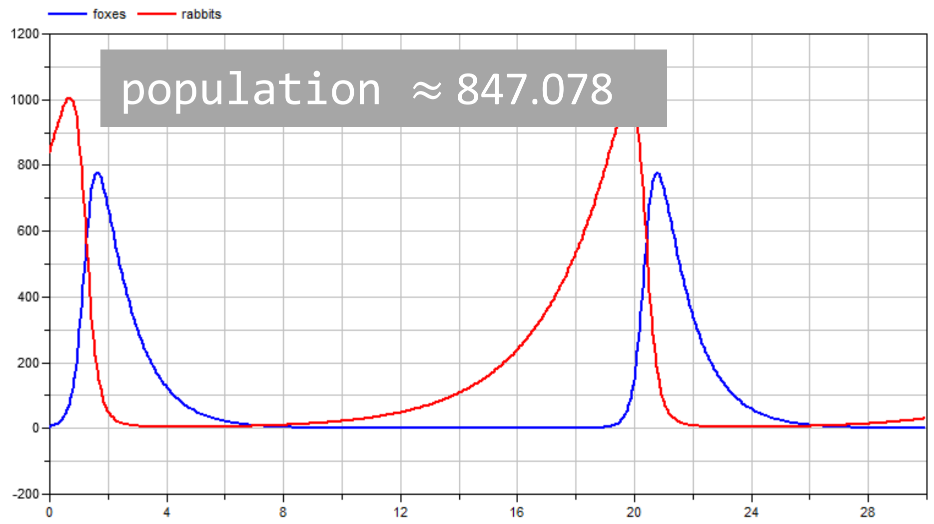
Numerical Method

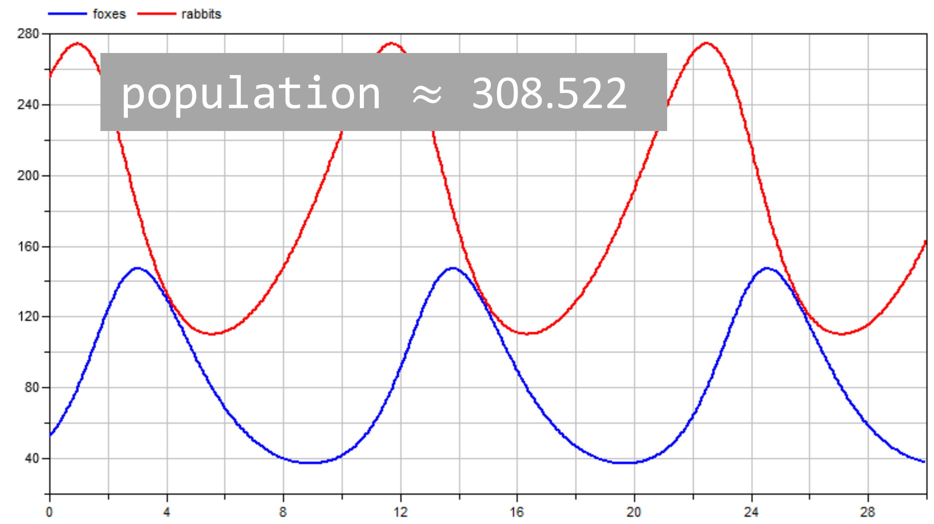
Example

```
model forest
  Real foxes;
  Real rabbits;
  Real population(start=350);
  Real value;
  […]

initial equation
  der(foxes) = 20;
  value = 11000;

equation
  der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
  der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
  population   = foxes+rabbits;
  value        = priceFox*foxes + priceRabbit*rabbits;
end forest;
```





Symbolic Method

Symbolic Method

Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Basic Approach

$$\begin{pmatrix}\underline{z}(t_0)\\ \underline{0}\end{pmatrix} = \begin{pmatrix} \underline{g}\left(\underline{\omega}(t_0), \underline{d}(t), \underline{p}^{fixed}, t\right) \\ \underline{h}\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{d}(t_0), \underline{p}^{fixed}, t_0\right)\end{pmatrix}$$

Involved Techniques

- matching
- sorting
- tearing
- …

Symbolic Method

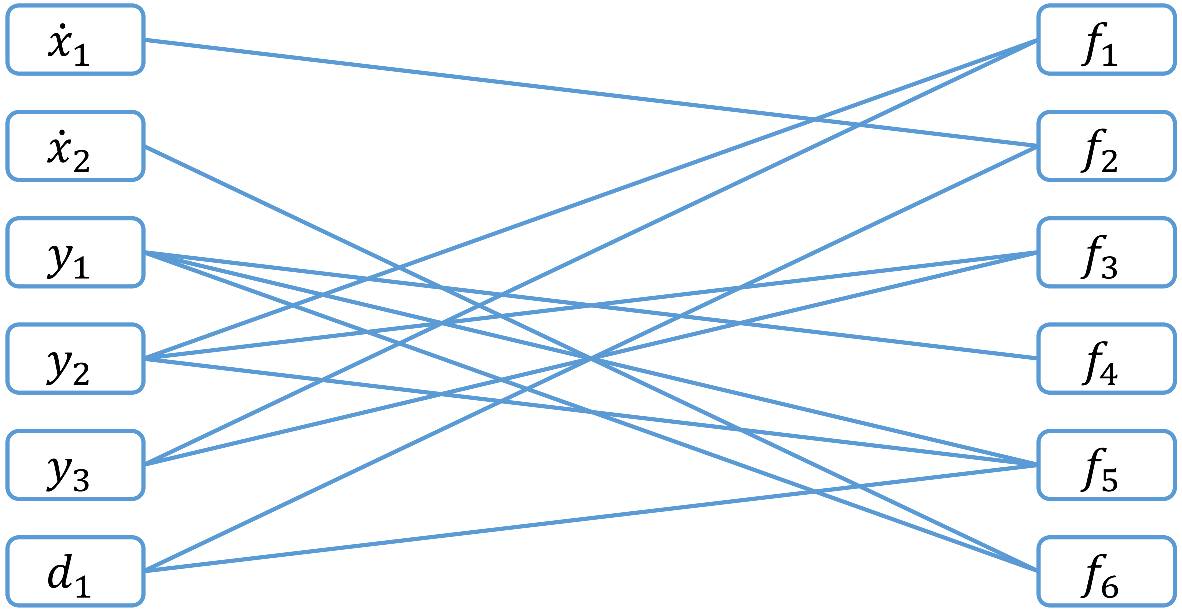
Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Matching (initial system)



Symbolic Method

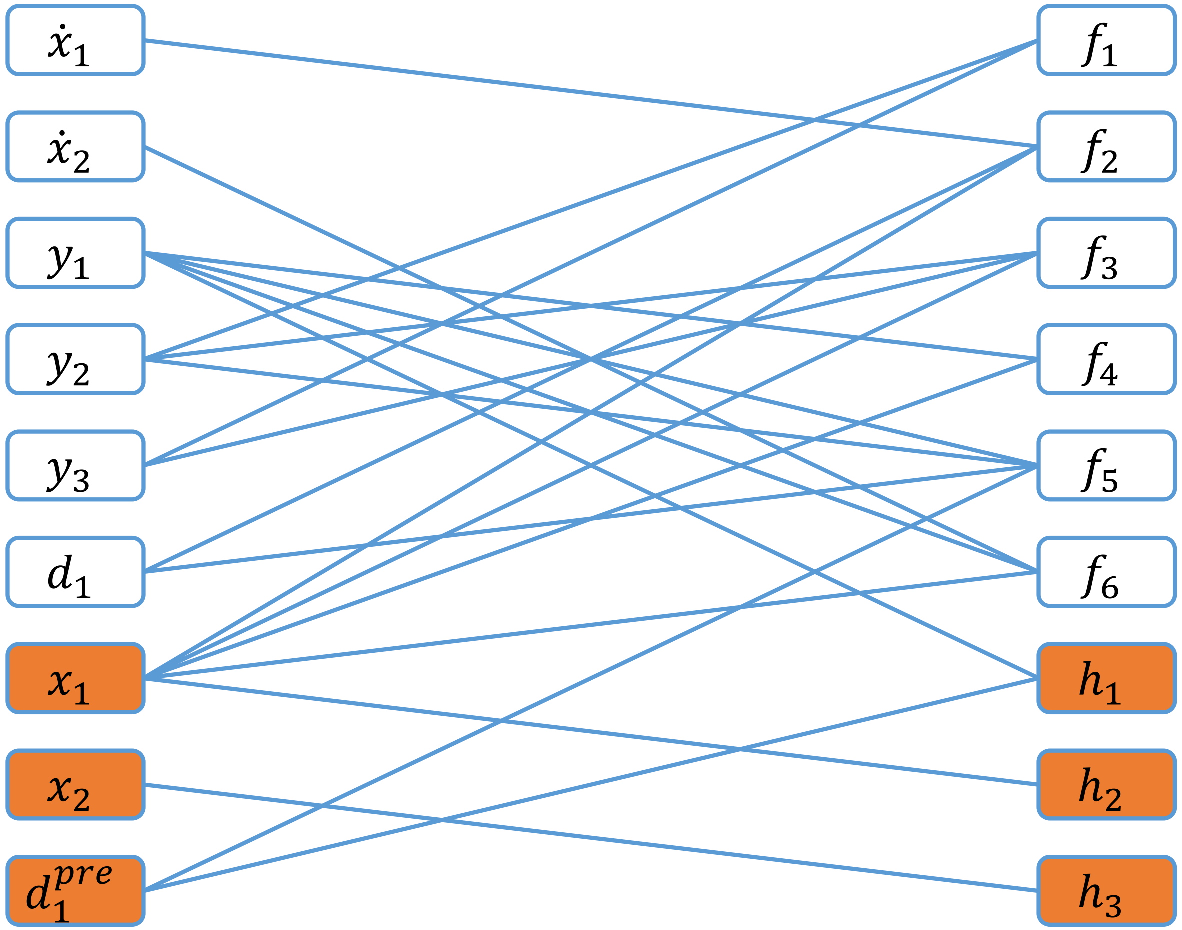
Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Matching (initial system)



Symbolic Method

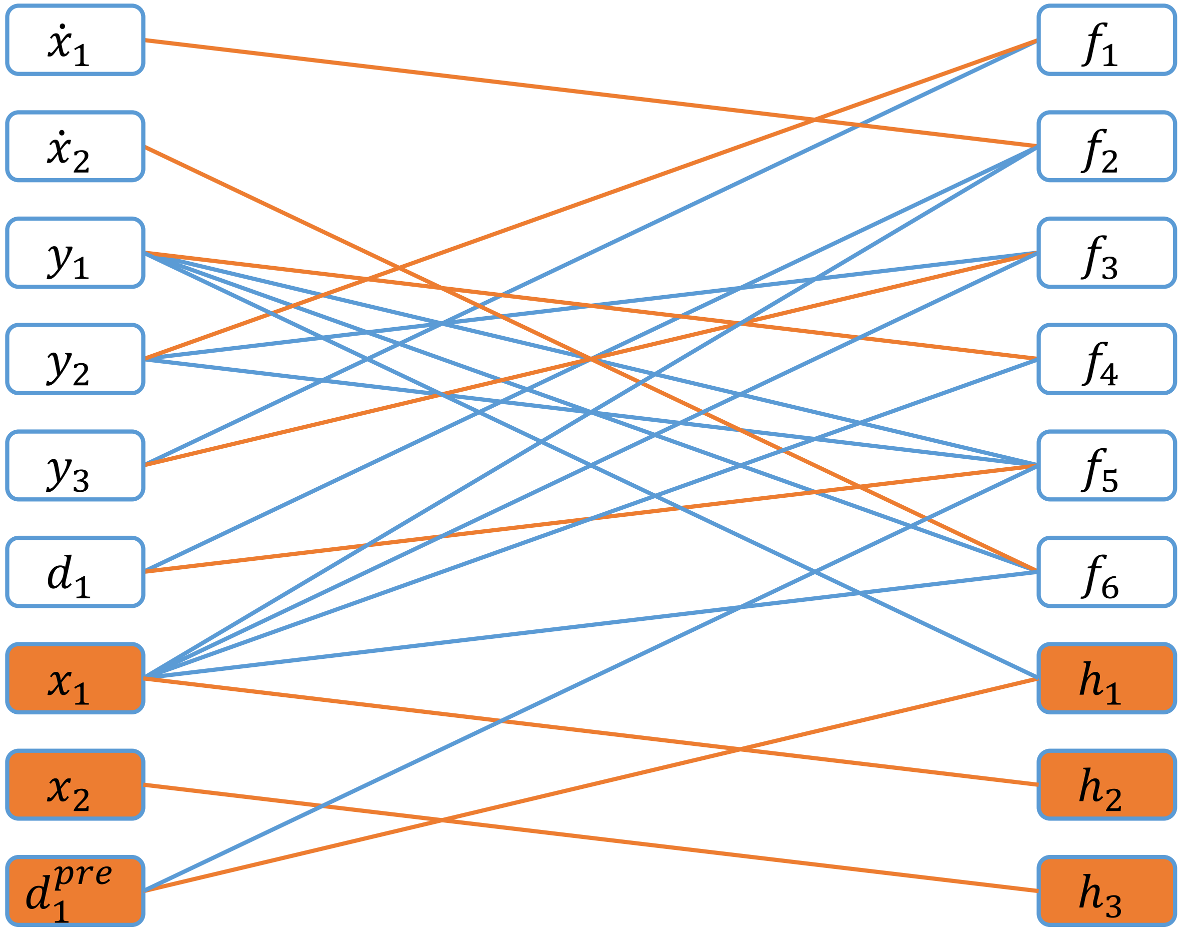
Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Matching (initial system)



Symbolic Method

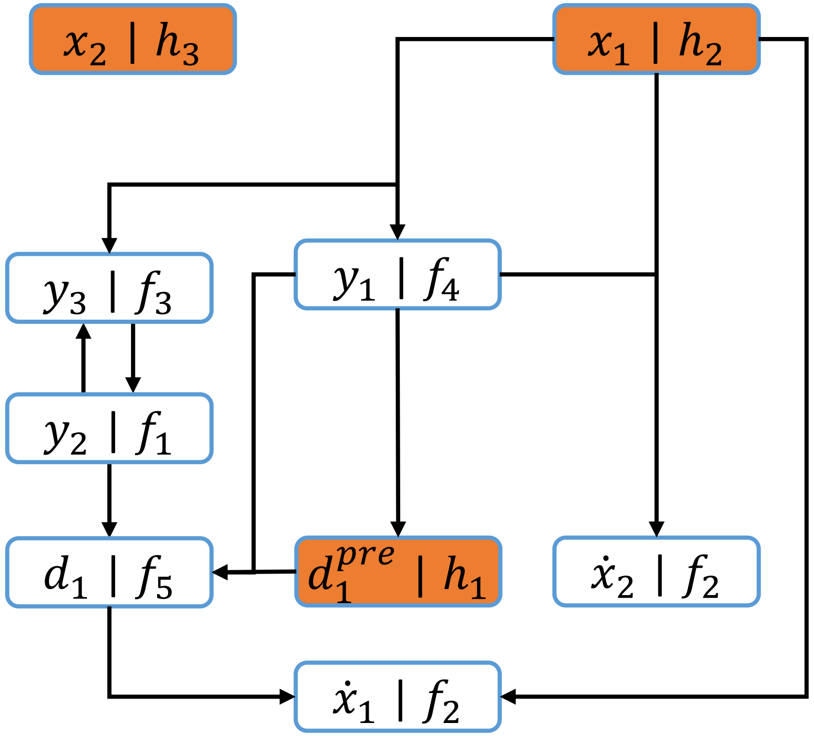
Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Strong Components



Symbolic Method

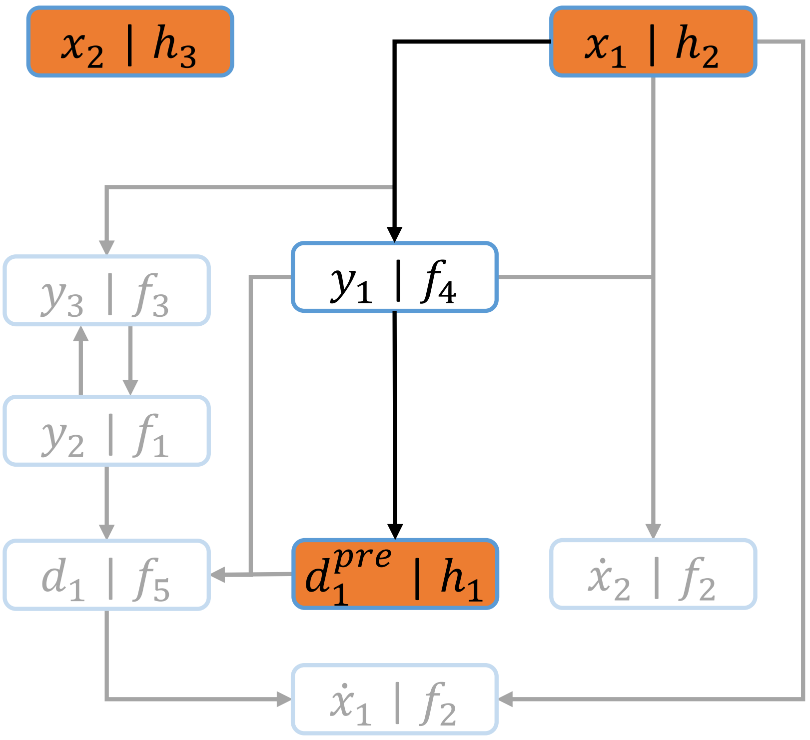
Example

```
    model MathRep
      Real x1(start=2.0, fixed=true),
           x2(start=4);
      Real y1, y2, y3(start=-1.5);
      Real d1;

    initial equation
h1    pre(d1) = -0.5 + y1;

    equation
f1    0 = -y2 + sin(y3);
f2    der(x1) = sqrt(x1) + time - d1;
f3    0 = x1 + y2 + y3 + 1;
f4    0 = x1 + y1 + x1*y1;
      when {initial(), sample(0.1, 0.1)} then
f5      d1 = pre(d1) - y1 + y2;
      end when;
f6    der(x2) = x1 + y1;
    end MathRep;
```

Strong Components



Conclusion and Outlook

Conclusion

Numeric Method

- handles over-constrained problems
- no full support for discrete variables
- Start Value Homotopy
- bad performance for real-world models

Symbolic Method

- no over-constrained problems
- full support for discrete variables
- no Start Value Homotopy
- good performance for real-world models

Conclusion

Numeric Method

- handles over-constrained problems
- no full support for discrete variables
- Start Value Homotopy
- bad performance for real-world models

Symbolic Method

- no over-constrained problems
- full support for discrete variables
- no Start Value Homotopy
- good performance for real-world models

Outlook

merge all advantages from both methods