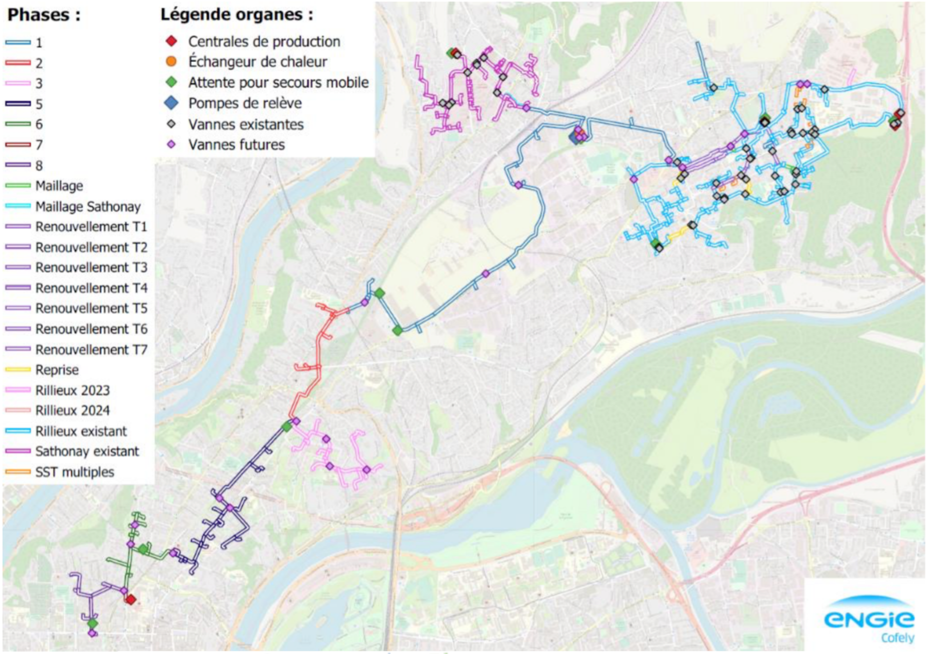
Modeling Urban Energy Networks with Modelica
The Scalability Challenge

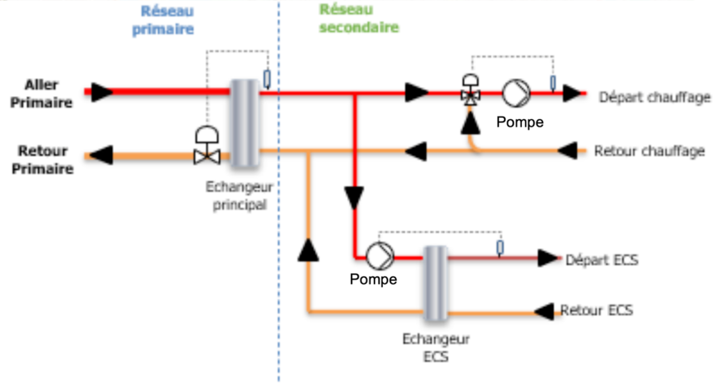
Benoît Caillaud, Mathias Malandain, Joan Thibault

<firstname.lastname@inria.fr>

City energy modelling: the case of urban heat networks

- Large networks comprising thousands of components : pipe sections; 2-/3-/4-way proportional, pressure safety, one-way & isolation valves, pumps, boilers, heat exchangers, storage & expansion tanks, …
- Non-linear laws everywhere: pressure drop vs. flow, enthalpy, viscosity, saturated steam pressure vs. temperature, …

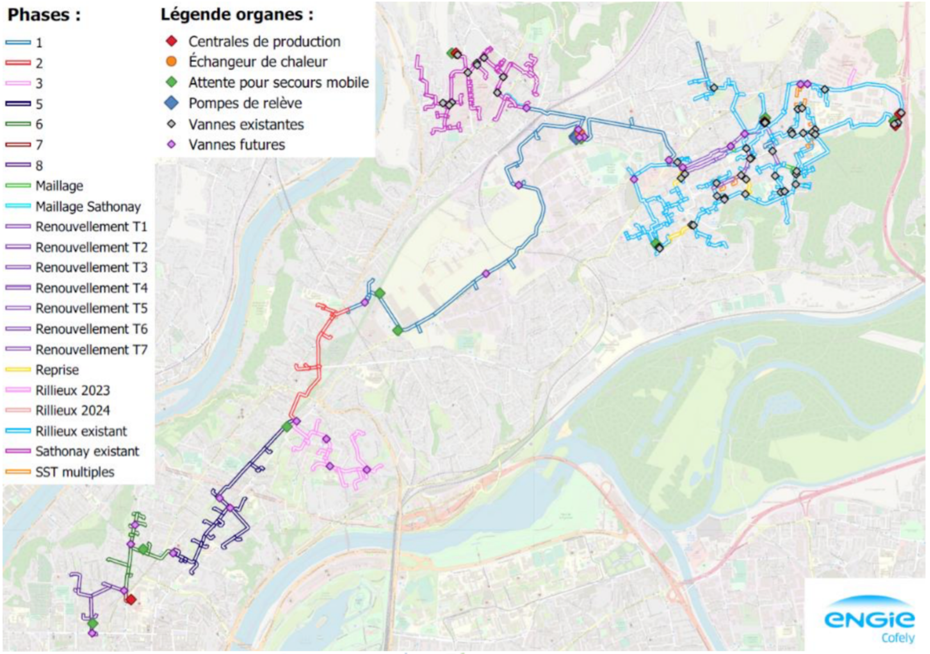


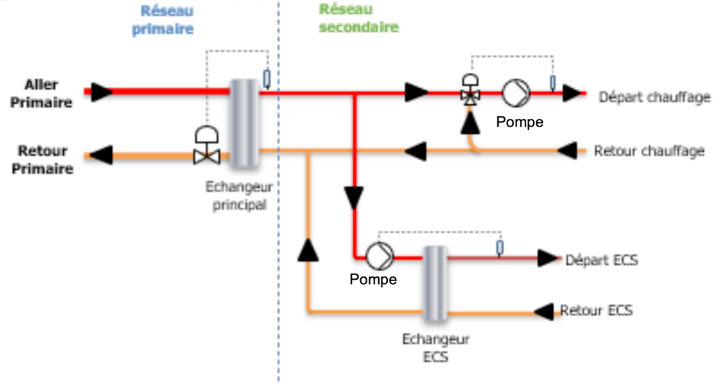


- Modelling as a state-space form model (with ODEs $x' = f(t,x)$) would be a daunting task
- Only the use of Differential Algebraic Equations (DAEs $g(t,x',x)=0$) enable a component-based modelling methodology (Modelica)
- Extremely large but sparse model: ≈120 sub-stations, ≈3.10^5 equations

City energy modelling: the case of urban heat networks

- Multimode (switched equations)
  - nonsmooth physics (typ. one-way valve)
  - many configurations & possible failures
- Modelica language allows multimode DAE systems (mDAE)
- SotA Modelica tools support only a limited, uncharacterized, subset of the language
- Failed to simulate whole heat network:
  - Time- varying structure
  - Workaround: stiff regularization ⇒ numerical inaccuracy, slow simulations
  - Consistant initialization is difficult





Component-based modelling with Modelica

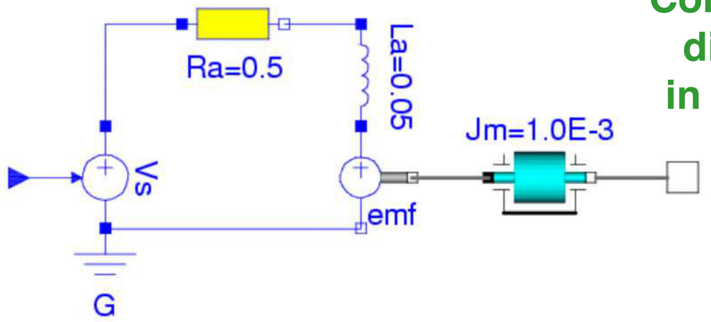
- Component-based modelling : DAEs rather than ODEs
  - Acausal components : differential + algebraic equations
  - Interconnections : algebraic equations (equal pressures + conservation of mass)

```
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;
```

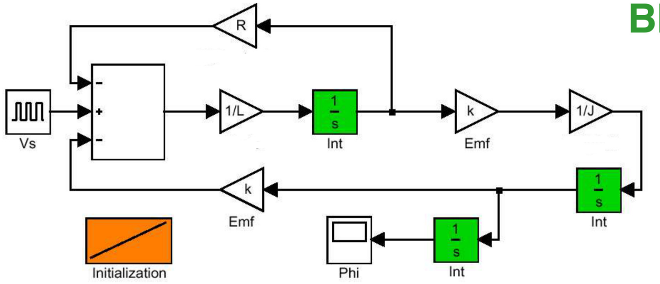
```
model Resistor
    package SIunits = Modelica.SIunits;
    parameter SIunits.Resistance R = 1;
    SIunits.Voltage v;
    ..Interfaces.PositivePin p;
    ..Interfaces.NegativePin n;
equation
    0 = p.i + n.i;
    v = p.v - n.v;
    v = R*p.i;
end Resistor;
```

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

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



Component diagram in Dymola

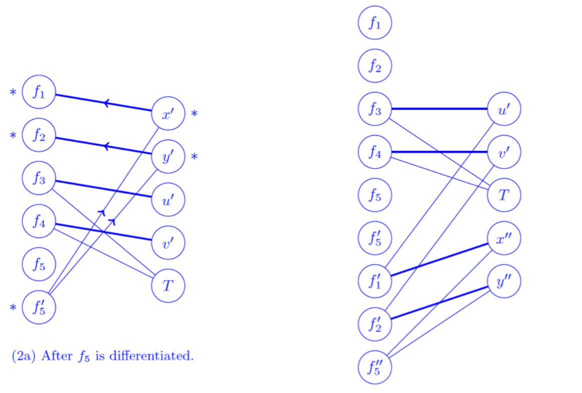


Block diagram in Simulink

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

Difficulties with multimode DAE systems

- Component-based modelling : DAE
  - Acausal components : differential + algebraic equations
  - Interconnections : algebraic equations (equal pressures + conservation of mass)

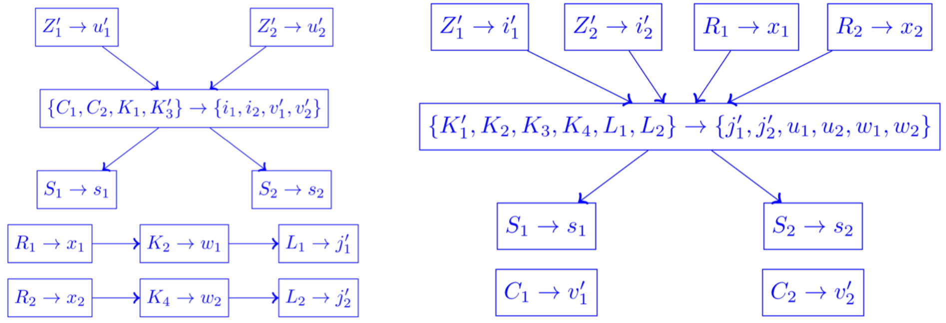


(2a) After f_5 is differentiated.

(2b) After f_1, f_2 and f_5' are differentiated.

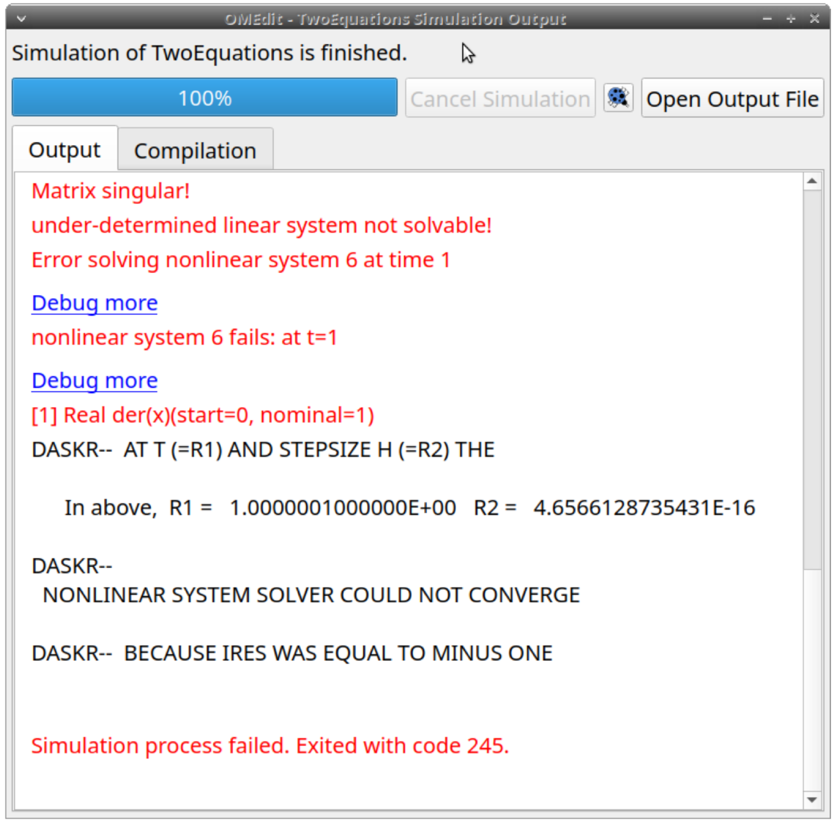
- Need for a Structural analysis (SA)
  - Compile-time index reduction + block triangular decomposition + static scheduling of equation blocks
  - Generation of efficient simulation code
  - Helps debugging models

- SA algorithms implemented in SotA Modelica tools not adapted to multimode systems
  - Designed for single mode DAEs
  - Ignore mode-dependencies
  - Generation of incorrect simulation code



Difficulties with multimode DAE systems

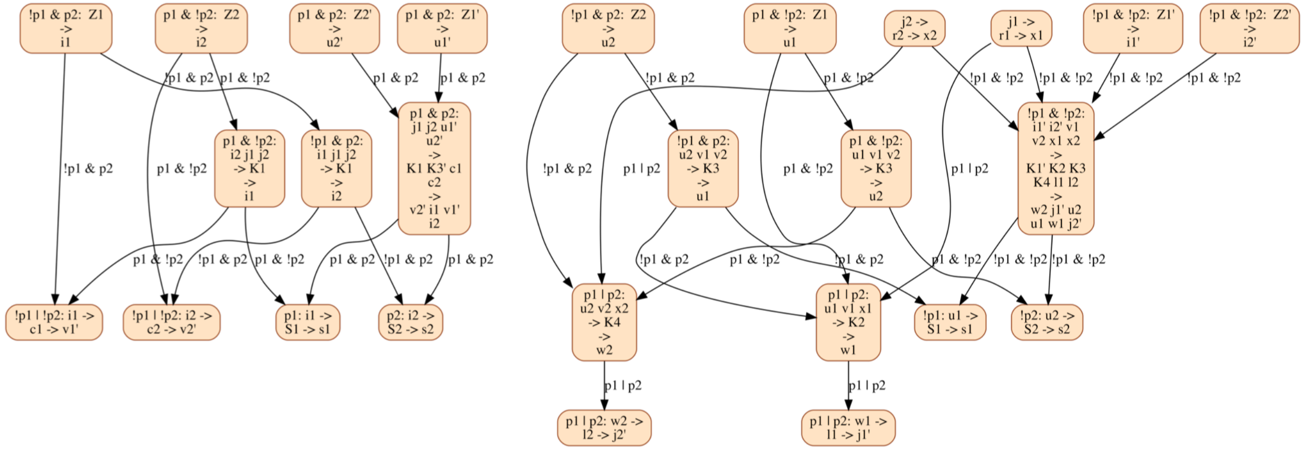
```
model TwoEquations
  Real x(start=0,fixed=true);
  Boolean p(start=false,fixed=true);
equation
  p = (x >= 1);
  1 = if p then x else der(x);
end TwoEquations;
```



- Runtime exception / Expected behavior at t=1 with OpenModelica (and all SotA Modelica tools)
- x’ is deemed to be the leading variable; second equation used to compute x’
- This equation is singular in x’ when p=True

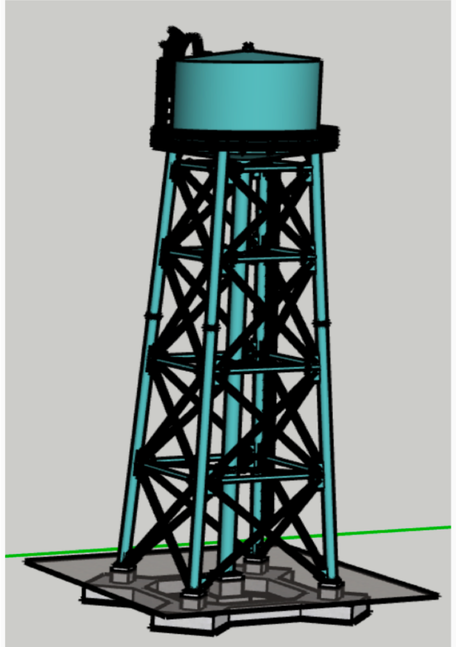
Multimode DAE Structural Analysis

- Design of a mDAE structural analysis algorithm *[Caillaud et al., 2020, HSCC]*
- Data structures adapted to the “combinatorial explosion” of modes
- Structural analysis of all modes “at once”

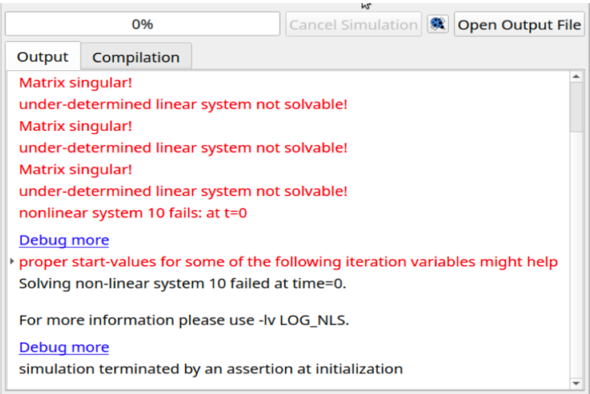


- Helps:
  - Generation of correct & efficient simulation code in all modes
  - Model debugging, thanks to precise, mode-dependent compile-time error-messages

The Water Tank model



```
model WaterTank
  Real t(start=0,fixed=true); // time (to define input flow)
  constant Real xmax = 1.0; // max water quantity
  constant Real xmin = 0.0; // min water quatity
  constant Real y0 = 6.667; // default output flow
  constant Real rho = 0.8; // input flow parameter
  Real x(start=0.5,fixed=true); // stored water mass
  Real yh; // output flow correction, when tank is full
  Real yl; // output flow correction, when tank is empty
  Real z; // input flow
  Real sh; // parameter of the full-tank CC
  Real sl; // parameter of the empty-tank CC
  Boolean bh(start=false,fixed=true); // mode full-tank
  Boolean bl(start=false,fixed=true); // mode empty-tank
  // bh and bl satisfy assertion not (bh and bl)
equation
  // input flow law
  /* et: */ der(t)=1;
  /* e1: */ z = rho*y0*(1+
          Modelica.Math.cos(2*Modelica.Constants.pi*t));
  // tank level differential equation
  /* e2: */ der(x) = z + yl - yh - y0;
  // Complementarity condition 0 <= xmax - x # yh >= 0
  bh = (sh >= 0);
  /* eh1: */ sh = if bh then yh else x - xmax;
  /* eh2: */ 0 = if bh then x - xmax else yh;
  // complementarity condition 0 <= x - xmin # yl >= 0
  bl = (sl >= 0);
  /* el1: */ sl = if bl then yl else xmin - x;
  /* el2: */ 0 = if bl then xmin - x else yl;
end WaterTank;
```



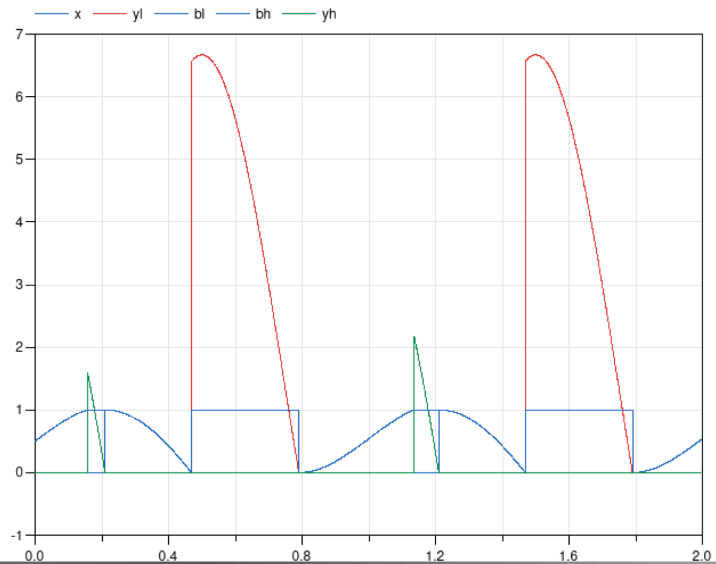
Simulation fails with OpenModelica v1.17.0 and Dymola 2021

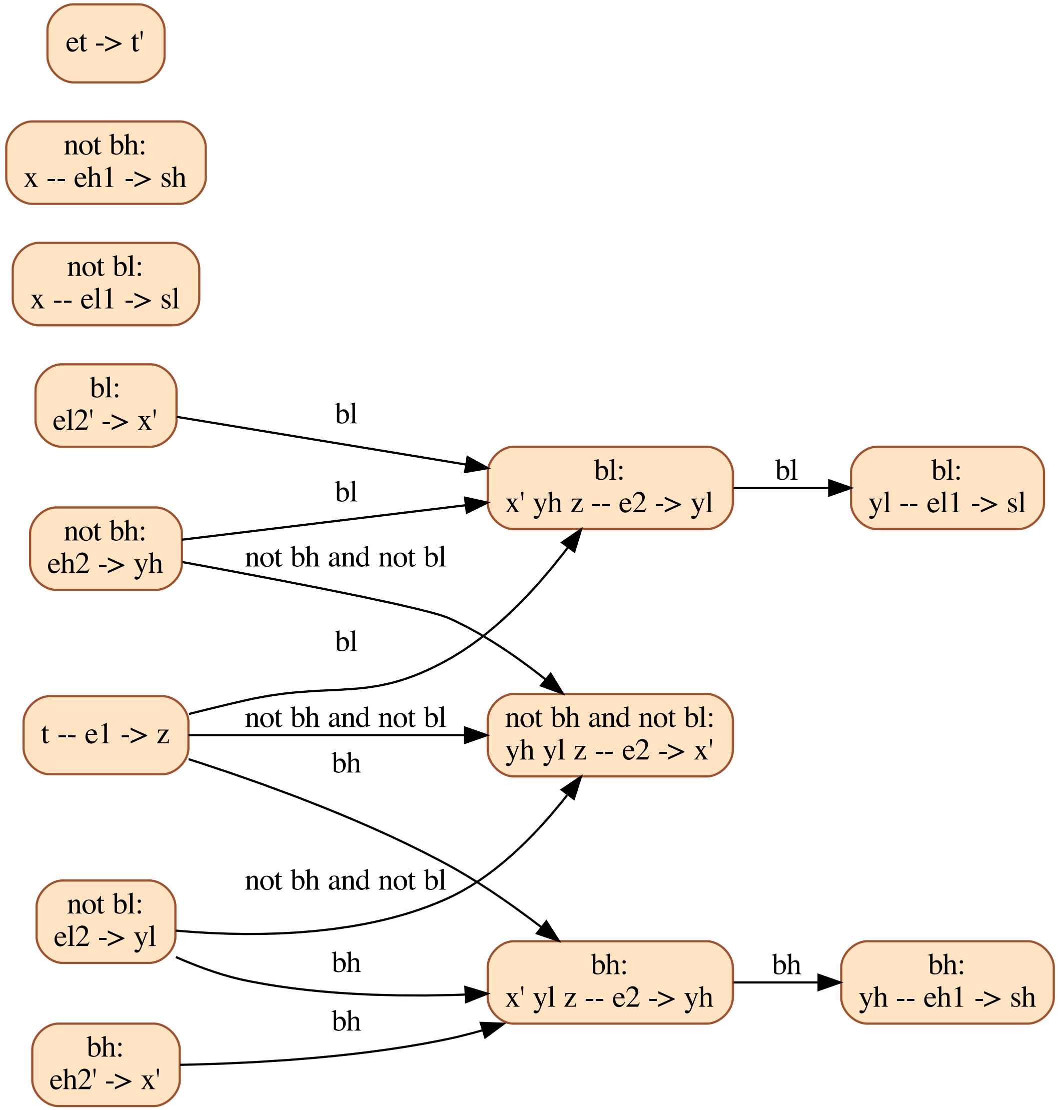
Input flow z and nominal output flow y defined as functions of time. Water quantity x. Complementarity system with three modes:

- Underflow ($x \leq x_{min}$): output flow $y - yl$, such that $yl \geq 0$ and $x \geq x_{min}$
- Nominal ($x_{min} < x < x_{max}$): nominal output flow y
- Overflow ($x_{max} \leq x$): output flow $y + yh$, such that $yh \geq 0$ and $x \leq x_{max}$

Multimode Structural Analysis of the Water Tank model

- Varying **structure** model
- Varying **structural differentiation index**
- Reduced Index Mode Independent Structure (RIMIS) : source to source transformation turning the model into a model handled correctly by SotA Modelica tools [Modelica'21]

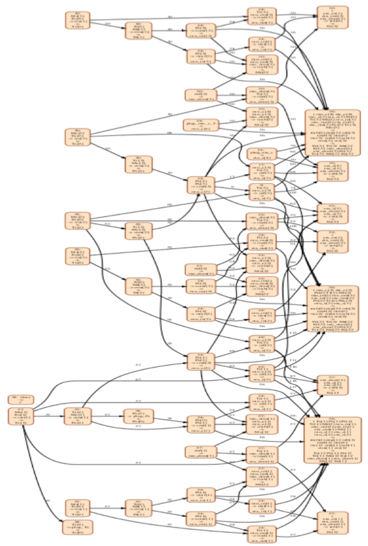




Implementation: IsamDAE

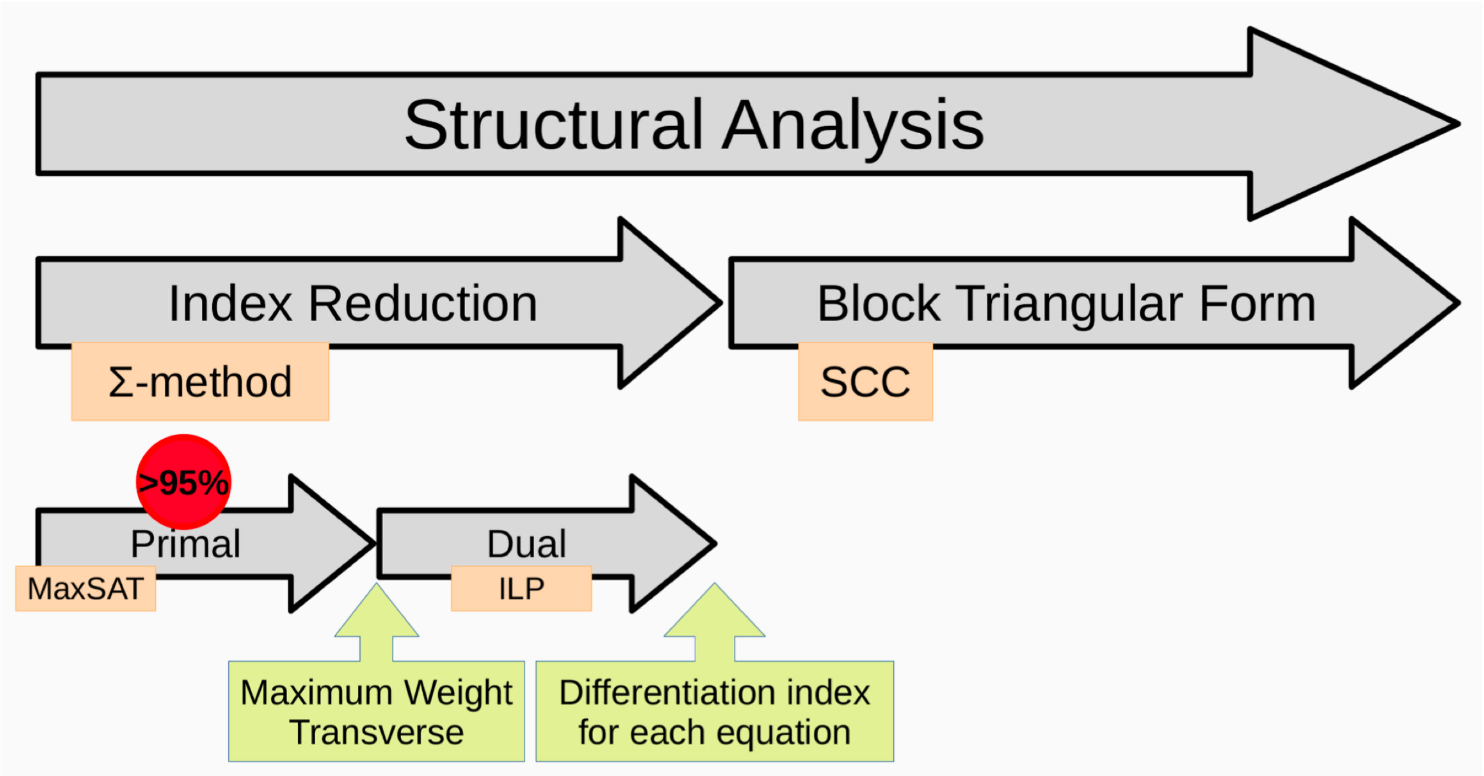
- Structural analysis of modes & consistent initialization of mDAEs
- Designed to be used at the heart of Modelica compilers
- Uses Binary Decision Diagrams (BDD)
  - Representation of the mode-dependent structure of mDAEs
  - Compositional structural analysis method: scalability
- ≈ 4 persons x year effort, 35000 LoC
- Experimental prototype, integration of research results: scalability, impulsive mode-changes, …
- Can be tested on the AllGo web platform: https://allgo18.inria.fr/apps/isamdae

```
equation
 // equations for rooms
 for i in 1:N loop
   if compressible then
     Mr[i] = Vr * rhoCompressible(Pr[i], Tr[i]);
   else
     Mr[i] = Vr * rho(Tr[i]);
   end if;
   mu_in[i] = flow_vent(Pin - Pr[i]);
   mu_out[i] = flow_vent(Pr[i] - Pout);
   der(Mr[i]) = mu_in[i] - mu_out[i];
   Er[i] = Mr[i] * enthalpy(Tr[i]);
   eta_in[i] = mu_in[i] * enthalpy(Tin);
   eta_out[i] = mu_out[i] * enthalpy(Tr[i]);
   der(Er[i]) = eta_in[i] - eta_out[i];
 end for;
```



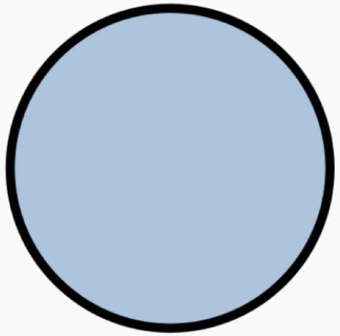
Highlights on mDAE Structural Analysis

- Index reduction
  - Generalization of J. Pryce's ∑-method
  - Primal problem : Maximum. Weight Transverse of a bipartite graph
  - Dual Problem : Least fixpoint computation on integer functions by iterative method
- Mode-dependent Block Triangular Decomposition
  - Mode-dependent strongly connected components
  - Least fixpoint computation on Boolean functions



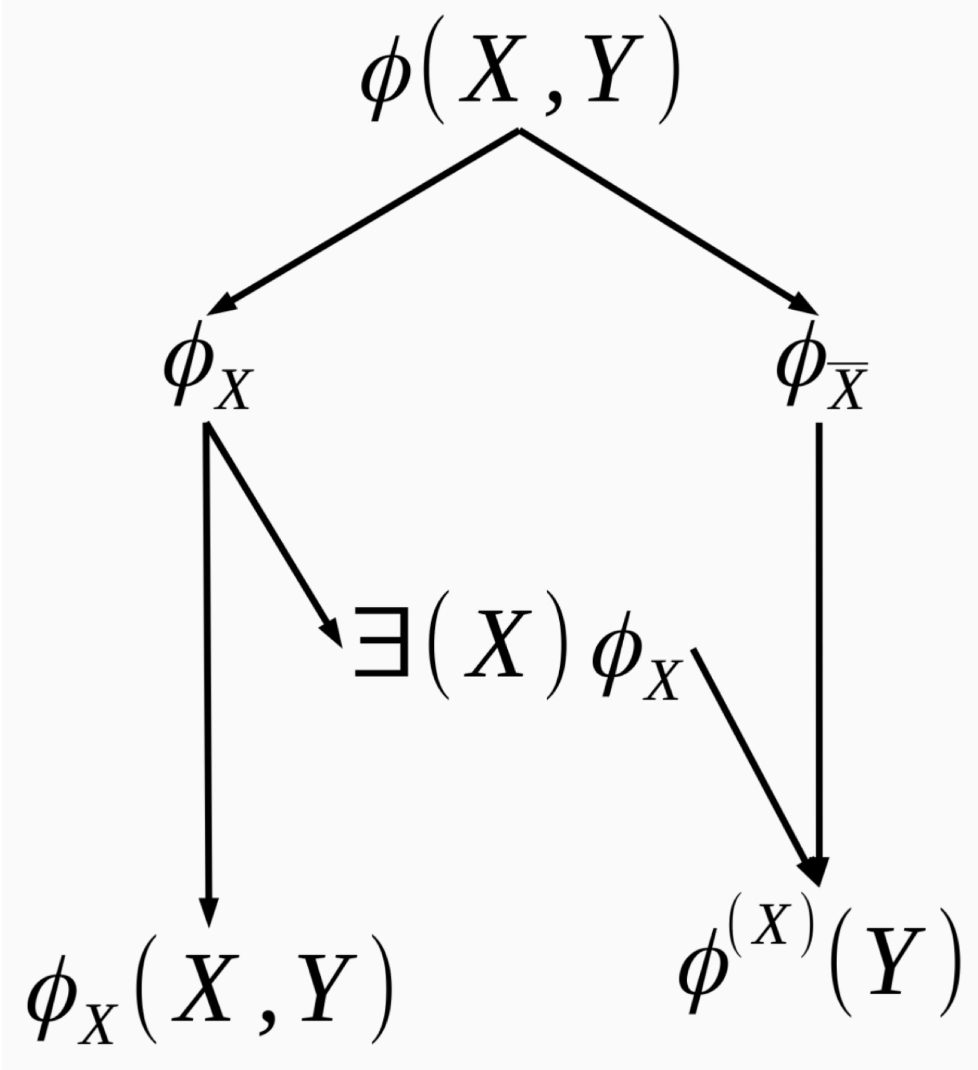
Reduced Block Triangular Decomposition (RBTF)

- Maximal Weight Transverse
  - MaxSAT problem
  - Model sparsity $\Longrightarrow$ sparse Boolean equations
- RBTF
  - Decomposition of system of Boolean equations : forward propagation
  - Solve locally
  - Combine partial solutions : backward propagation
- WAP decomposition heuristics
  - Based on upper bound estimation of BDD sizes
  - Tree-width problem



Highlights on RBTF : forward decomposition

- Maximal Weight Transverse
  - MaxSAT problem
  - Model sparsity $\Longrightarrow$ sparse Boolean equations
- RBTF
  - Decomposition of system of Boolean equations : forward propagation
  - Solve locally
  - Combine partial solutions : backward propagation
- WAP decomposition heuristics
  - Based on upper bound estimation of BDD sizes
  - Tree-width problem

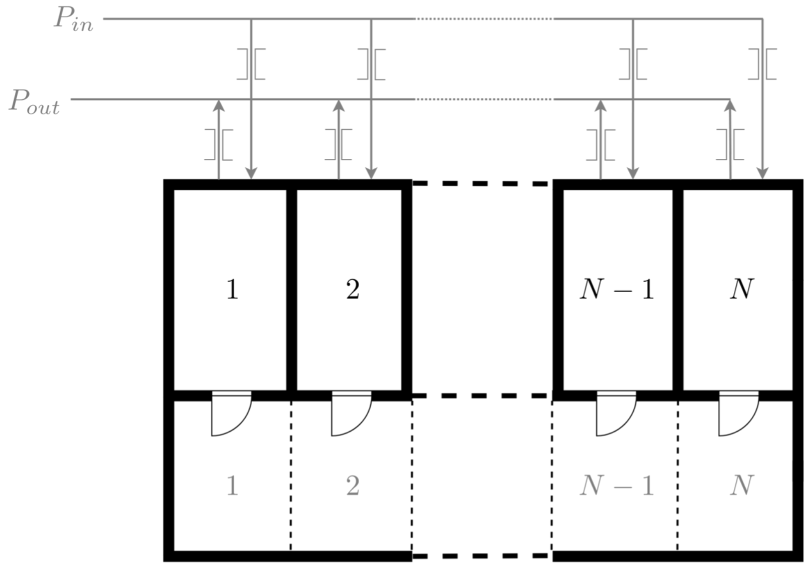


Benchmarking : thermal models of buildings

- Open/closed doors ⇒ multimode
- Varying structure

```
equation
 // equations for rooms
 for i in 1:N loop
   if compressible then
     Mr[i] = Vr * rhoCompressible(Pr[i], Tr[i]);
   else
     Mr[i] = Vr * rho(Tr[i]);
   end if;
   mu_in[i] = flow_vent(Pin - Pr[i]);
   mu_out[i] = flow_vent(Pr[i] - Pout);
   der(Mr[i]) = mu_in[i] - mu_out[i];
   Er[i] = Mr[i] * enthalpy(Tr[i]);
   eta_in[i] = mu_in[i] * enthalpy(Tin);
   eta_out[i] = mu_out[i] * enthalpy(Tr[i]);
   der(Er[i]) = eta_in[i] - eta_out[i];
 end for;
```

```
if open[i] then
  Pr[i] = Pc[i];
else
  mu_door[i] = 0;
end if;
```



- Failed simulations with Modelica tools

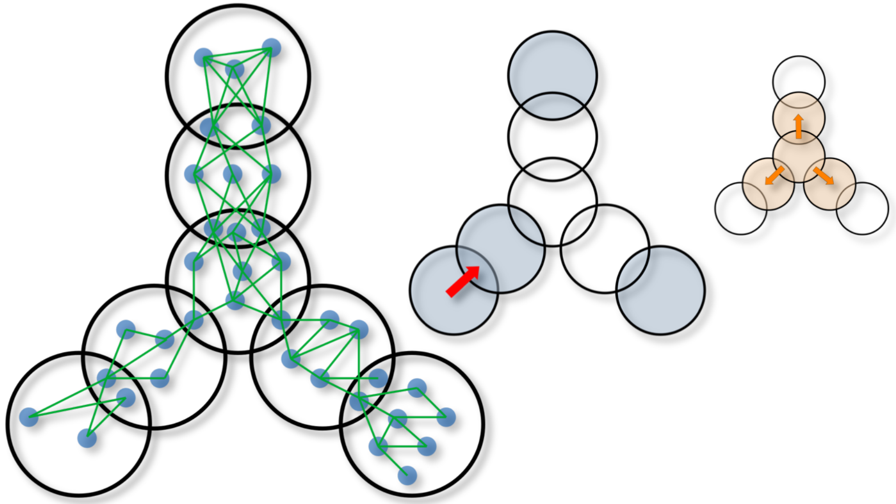
```
The following error was detected at time: 0
     Error: Singular inconsistent scalar system for
     mu_door[3] = ( -(if open[3] then Pr[3]-Pc[3] else
     0.0))/((if open[3] then 0.0 else 1.0)) = -159141/0
```

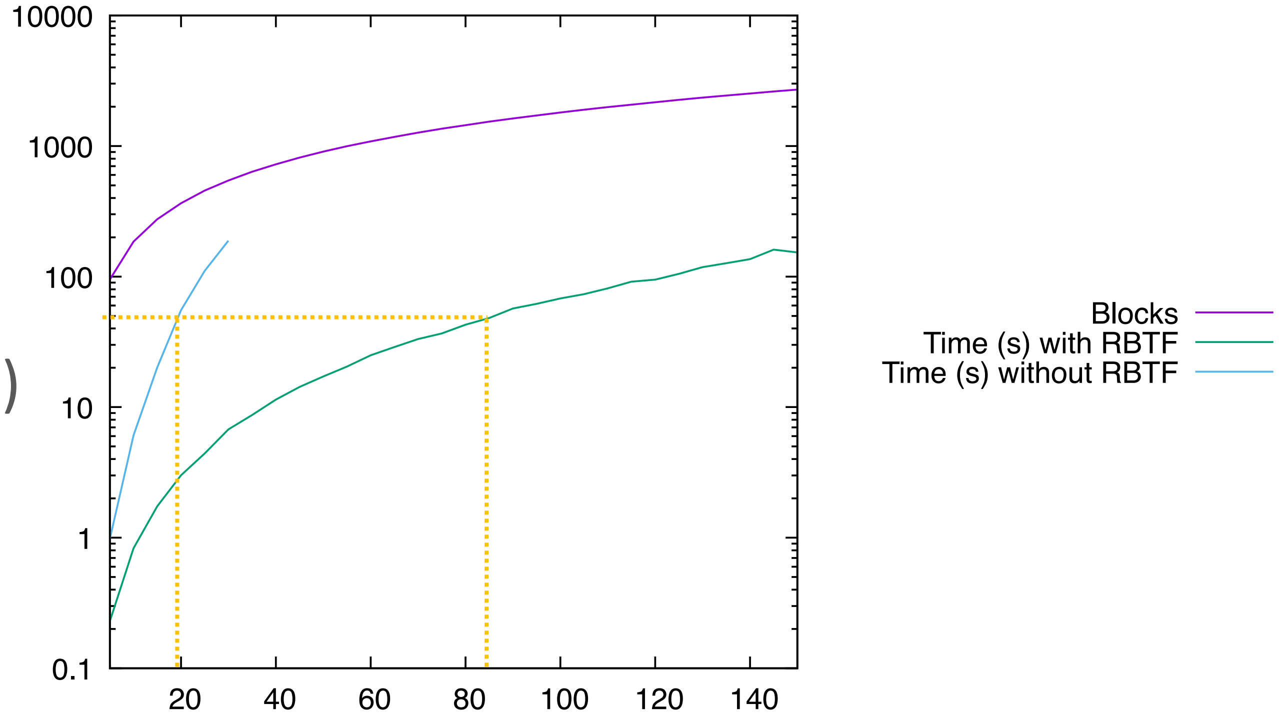
Benchmarking : thermal models of buildings

- Mode combinatorics :

  N rooms → *6^N/2* modes

- Empirical time/memory complexity : $O(N^2)$
- Thanks to the compositional method (RBTF) implemented in IsamDAE





> 200 components

> 3000 equations

> 10^{150} modes

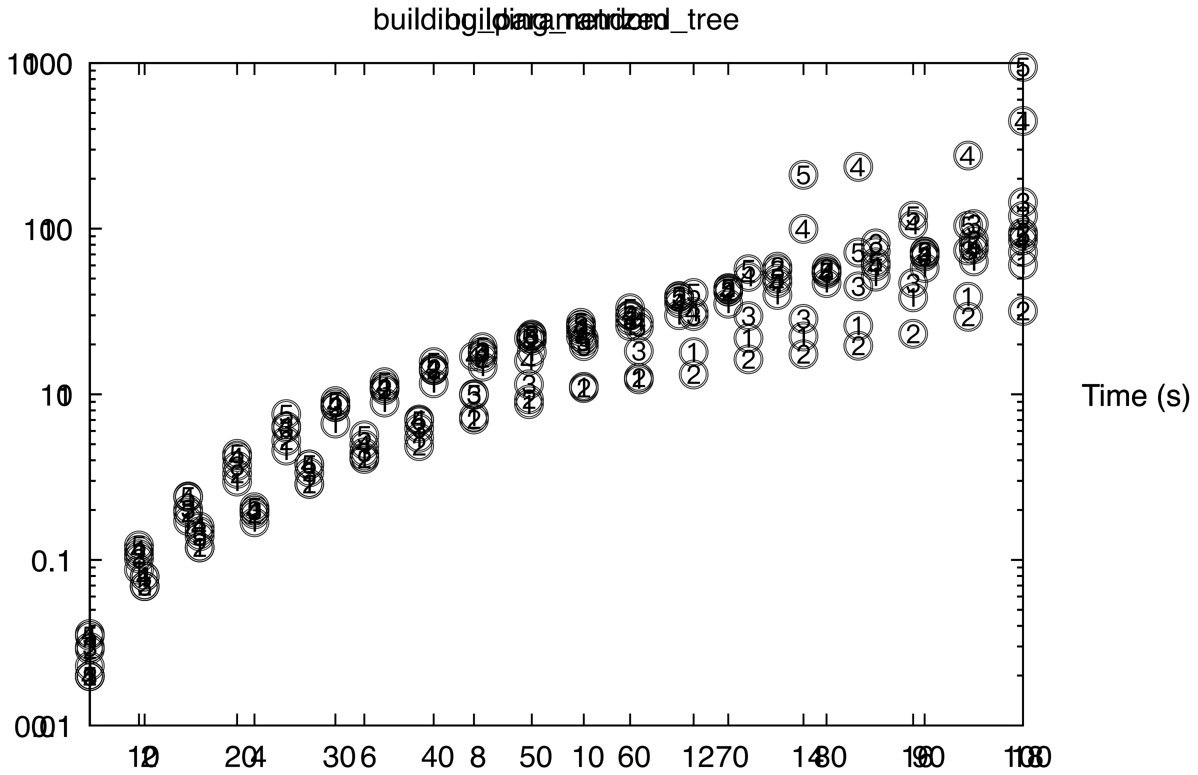
Benchmarking : varying topologies

- Geometry of physical system impacts

  Logical topology model

  & Model sparsity

- Experimented with various topologies
  - Trees of varying degree
  - Ring, grid
  - Sparse random graphs
- RBTF scales up : $O(N^2)$ except for grids



Perspectives

- Complete compilation chain for multimodes DAEs…
  - Structural analysis of modes and consistent initialization: done
  - Structural analysis of (impulsive) mode-changes: published, to be implemented

- ...supporting digital twins of large-scale cyberphysical systems...
  - Modular structural analysis method
  - ”per component/subsystem” approach : better suited to component-based modelling

- ...ready to be used in Modelica tools
  - Redesign of Modelica compiler backends, to handle mode-dependent schedulings
  - Extensions of the Modelica language (varying dimension, mode-dependent initialization, dynamic reconfiguration, …)

ModeliScale

Passer Modelica à l'échelle pour la modélisation et la simulation des grands systèmes cyber-physiques énergétiques industriels, pour modéliser leurs nouvelles architectures induites par la loi de transition énergétique