

Higher-Order Non-Causal Modelling and Simulation of Structurally Dynamic Systems

George Giorgidze Henrik Nilsson

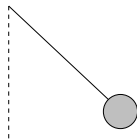
Functional Programming Laboratory
School of Computer Science
University of Nottingham

7th International Modelica Conference
Como, Italy
2009 Sep 21

Structurally Dynamic Systems

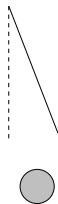
- ▶ Models whose equational description change over time are called *structurally dynamic*
- ▶ Each structural configuration is known as a *mode* of operation

Mode 1 - Pendulum



$$\begin{aligned}\frac{dx}{dt} &= l * \sin(\varphi) & v_x &= \frac{dx}{dt} \\ \frac{dy}{dt} &= l * \cos(\varphi) & v_y &= \frac{dy}{dt} \\ \frac{d^2\varphi}{dt^2} + \frac{g}{l} * \sin(\varphi) &= 0\end{aligned}$$

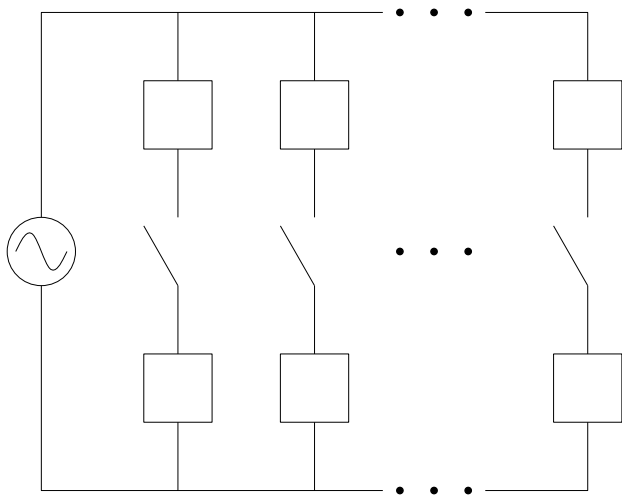
Mode 2 - Free Fall



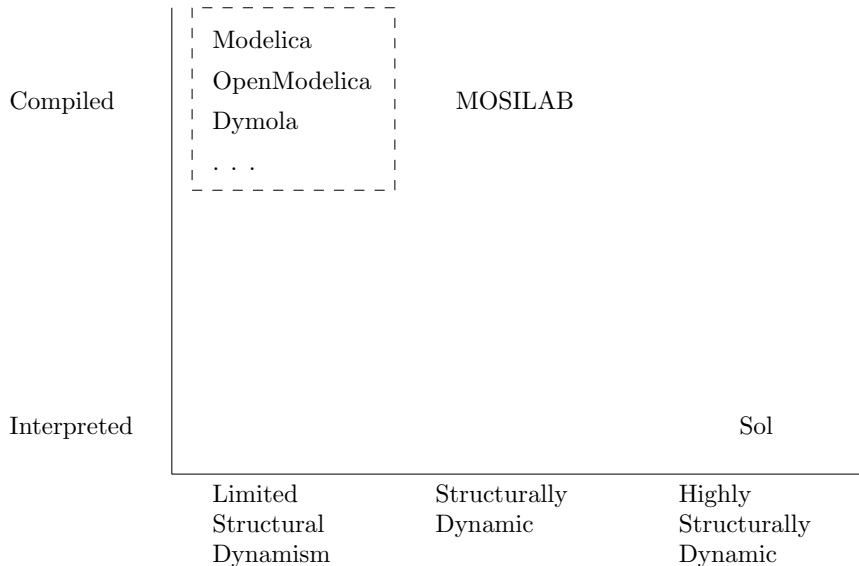
$$\begin{aligned}\frac{dx}{dt} &= v_x & \frac{dv_x}{dt} &= 0 \\ \frac{dy}{dt} &= v_y & \frac{dv_y}{dt} &= -g\end{aligned}$$

Highly Structurally Dynamic Systems

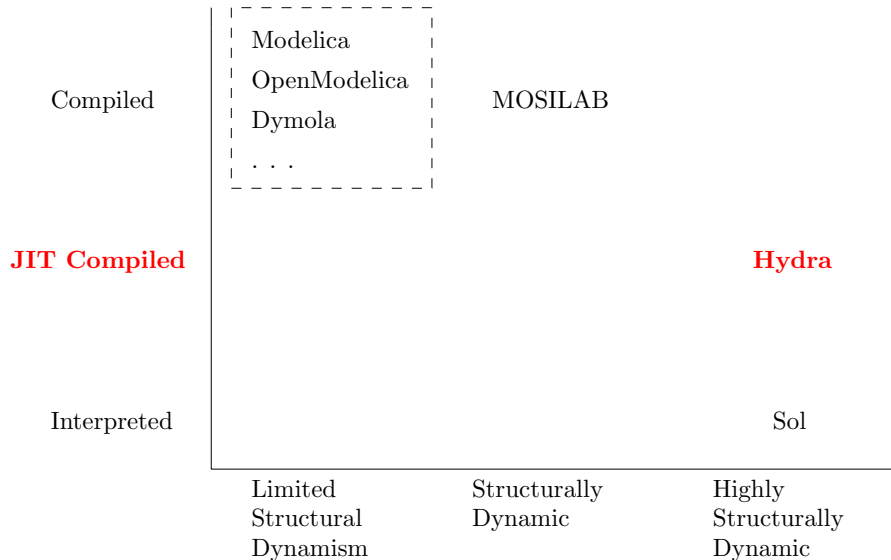
- ▶ Systems where the number of modes is too large or even a priori unbounded are called *highly structurally dynamic*



Structural Dynamism in Modelica-like Languages



This Talk



Functional Hybrid Modelling (FHM) and Hydra

- ▶ Functional Hybrid Modelling (FHM)
 - ▶ Identify core concepts of non-causal and hybrid modelling
 - ▶ Embed models as first-class entities in a declarative host language
- ▶ Hydra - FHM language we are currently working on

Paper and rest of the talk ...

- ▶ ***Higher-order modelling as an expressive language feature***
 - ▶ *Models* (think Modelica class) are first class entities (just like Ints or Booleans)
 - ▶ Programmatically manipulate models
 - ▶ Programmatically manipulate collections of models
 - ▶ **Compute models at run-time (structural dynamism)**

Read the paper

- ▶ **Novel approach to the implementation of non-causal modelling languages using just-in-time (JIT) code generation**

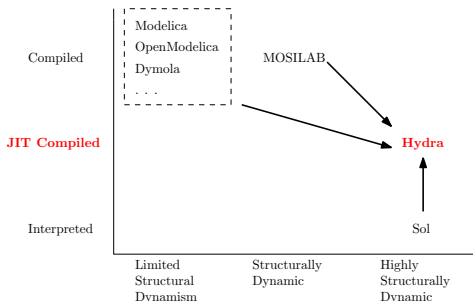
Rest of the talk

Is this useful for Modelica?

Is this useful for Modelica?

Novel implementation approach using JIT code generation

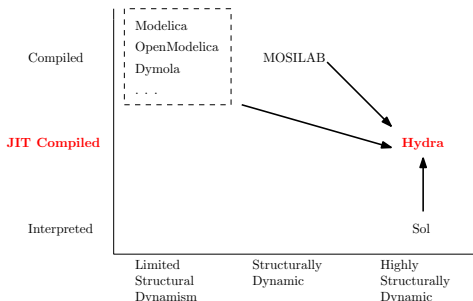
- ▶ Lift the restrictions that are associated with pre-simulation compilation of **Modelica** code
- ▶ Enable **MOSILAB** to handle highly structurally dynamic systems
- ▶ Enable **Sol** to target high-end simulation tasks
- ▶ Enable structural dynamism in **Modelling Kernel Language (MKL)** a possible core language for Modelica (MKL supports HOM)



Is this useful for Modelica?

Novel implementation approach using JIT code generation

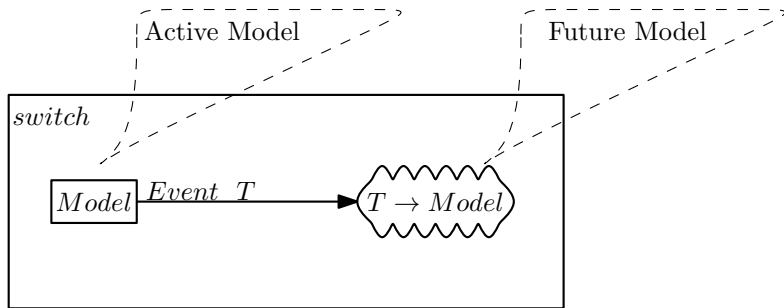
- ▶ Lift the restrictions that are associated with pre-simulation compilation of **Modelica** code
- ▶ Enable **MOSILAB** to handle highly structurally dynamic systems
- ▶ Enable **Sol** to target high-end simulation tasks
- ▶ Enable structural dynamism in **Modelling Kernel Language (MKL)** a possible core language for Modelica (MKL supports HOM)



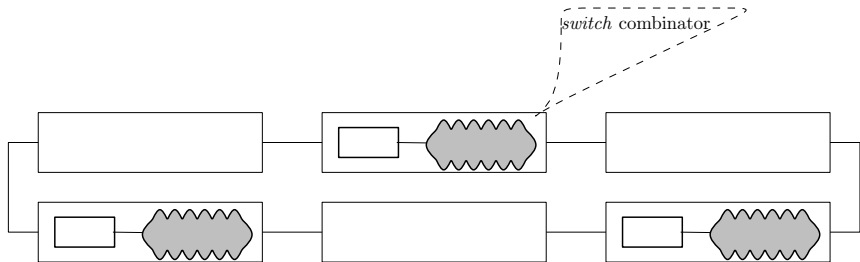
Higher-order modelling (future versions of Modelica language?)

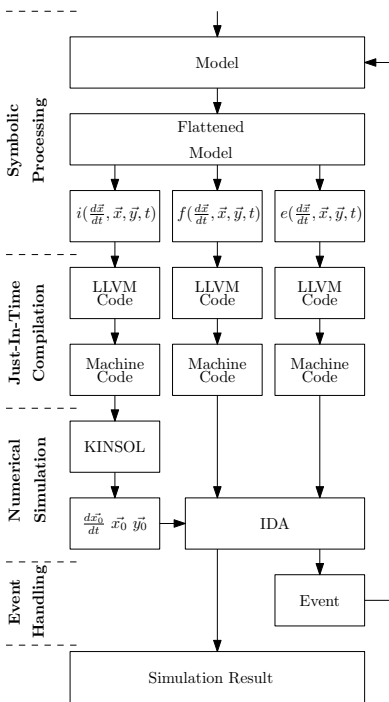
Structural Dynamism

Higher-order *switch* combinator

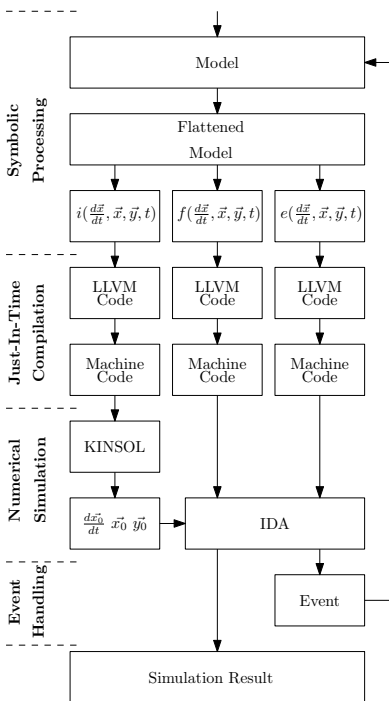


Structural Dynamism



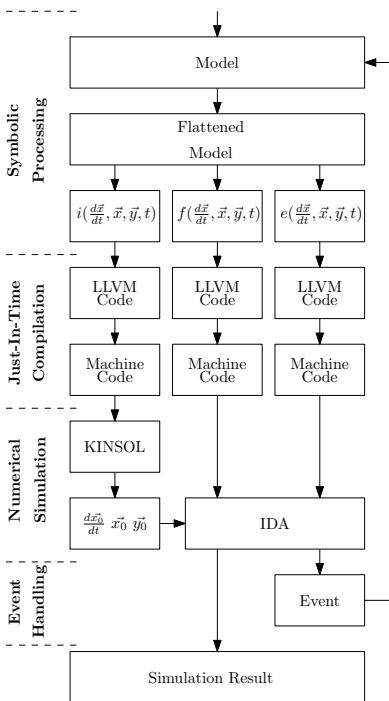


Simulation run-time system of Hydra



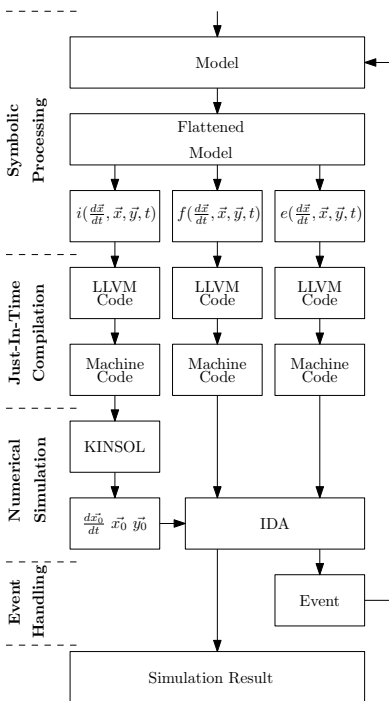
Symbolic Processing

- ▶ Flatten
- ▶ Transform into a mathematical representation suitable for numerical simulation



Low Level Virtual Machine (LLVM)

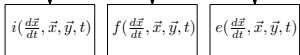
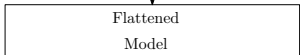
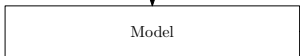
- ▶ Portable compiler target
- ▶ Flexible execution engine
- ▶ Efficient code generator
- ▶ Memory management



SUNDIALS

- ▶ SUite of Nonlinear and Differential/ALgebraic equation Solvers
- ▶ KINSOL: nonlinear algebraic equation systems solver
- ▶ IDA: differential algebraic equation systems solver

Symbolic Processing



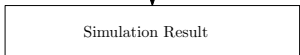
Just-In-Time Compilation



Numerical Simulation

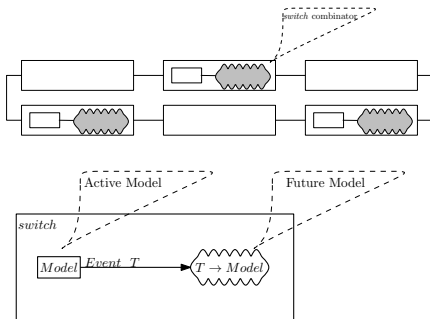


Event Handling



Event Handling

- ▶ Evaluate events
- ▶ Compute a new model



Performance

	Pendulum $t \in [0, 10)$		Free Fall $t \in [10, 20]$	
	CPU Time		CPU Time	
	s	%	s	%
Symbolic Processing	0.0001	0.2	0.0000	0.0
JIT Compilation	0.0110	18.0	0.0077	9.1
Numerical Simulation	0.0500	81.8	0.0767	90.9
Event Handling	0.0000	0.0	-	-
Total	0.0611	100.0	0.0844	100.0

Table: Time profile of the breaking pendulum simulation

Performance

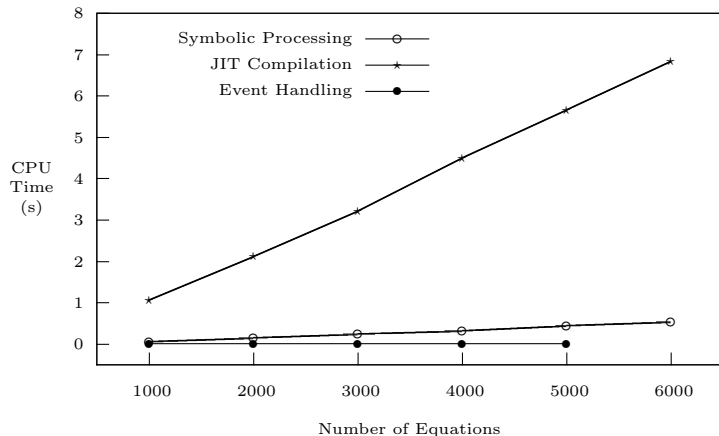
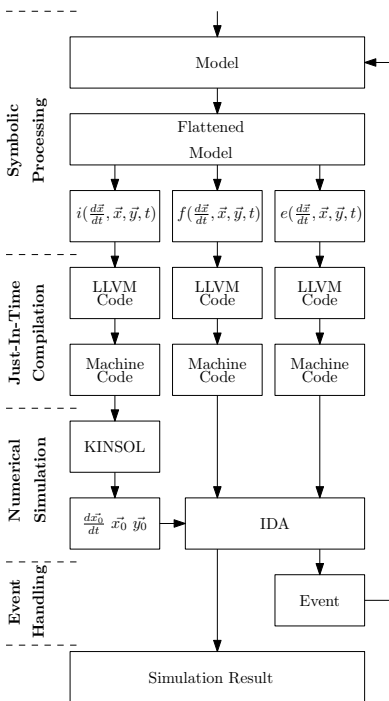


Figure: Plot demonstrating how CPU time spent on mode switches grows as number of equations increase in structurally dynamic RLC circuit simulation



Future Work

- ▶ Reuse generated code
- ▶ Take advantage of multi-core architectures
- ▶ Mixed interpreter and JIT approach

Conclusions

Novel approach to the implementation of non-causal modelling languages

- ▶ Efficient alternative to interpreted implementations
- ▶ Lifts restrictions of pre-simulation compilation

We used higher-order modelling provided by FHM

- ▶ Expressive language feature for structural dynamism

Read the paper and download the implementation (under the open source BSD license)

- ▶ <http://www.cs.nott.ac.uk/~ggg/>