Towards a Formal Semantics for FHM, Part I

FPL Away Days 2011

Henrik Nilsson

Joint work with Joey Capper

School of Computer Science

University of Nottingham

Hybrid Systems

Hybrid system: dynamical system with both discrete and continuous components.

Hybrid Systems

Hybrid system: dynamical system with both discrete and continuous components.

- Systems that inherently are hybrid; e.g., an automobile engine with digitally controlled fuel injection.

Hybrid Systems

Hybrid system: dynamical system with both discrete and continuous components.

- Systems that inherently are hybrid; e.g., an automobile engine with digitally controlled fuel injection.
- Models of continuous systems where simplifying assumptions leads to a hybrid formulation; e.g. ideal diode, bouncing ball.

Hybrid Automata (1)

Hybrid Automata: Standard approach for semantics of hybrid systems:

> Thomas A. Henzinger. The Theory of Hybrid Automata. In *Logic in Computer Science (LICS), 1996.*

Hybrid Automata (1)

Hybrid Automata: Standard approach for semantics of hybrid systems:

> Thomas A. Henzinger. The Theory of Hybrid Automata. In *Logic in Computer Science (LICS), 1996.*

- ***Variables***: finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables
  - $\dot{X}$ denotes first derivatives
  - X' denotes values after discrete change.

Hybrid Automata (2)

- ***Control graph***: ***finite*** directed multigraph (V, E);
  - vertices V called ***control modes***
  - edges E called ***control switches***

Hybrid Automata (2)

- ***Control graph***: ***finite*** directed multigraph (V, E);
  - vertices V called ***control modes***
  - edges E called ***control switches***
- ***Initial, invariant, flow conditions***: vertex labelling functions assigning predicate over X, X, and $X \cup \dot{X}$ respectively to each control mode $v \in V$

Hybrid Automata (2)

- ***Control graph***: ***finite*** directed multigraph (V, E);
  - vertices V called ***control modes***
  - edges E called ***control switches***
- ***Initial, invariant, flow conditions***: vertex labelling functions assigning predicate over X, X, and $X \cup \dot{X}$ respectively to each control mode $v \in V$
- ***Jump condition***: edge labelling function assigning predicate over $X \cup X'$ to each control switch $e \in E$

Hybrid Automata (3)

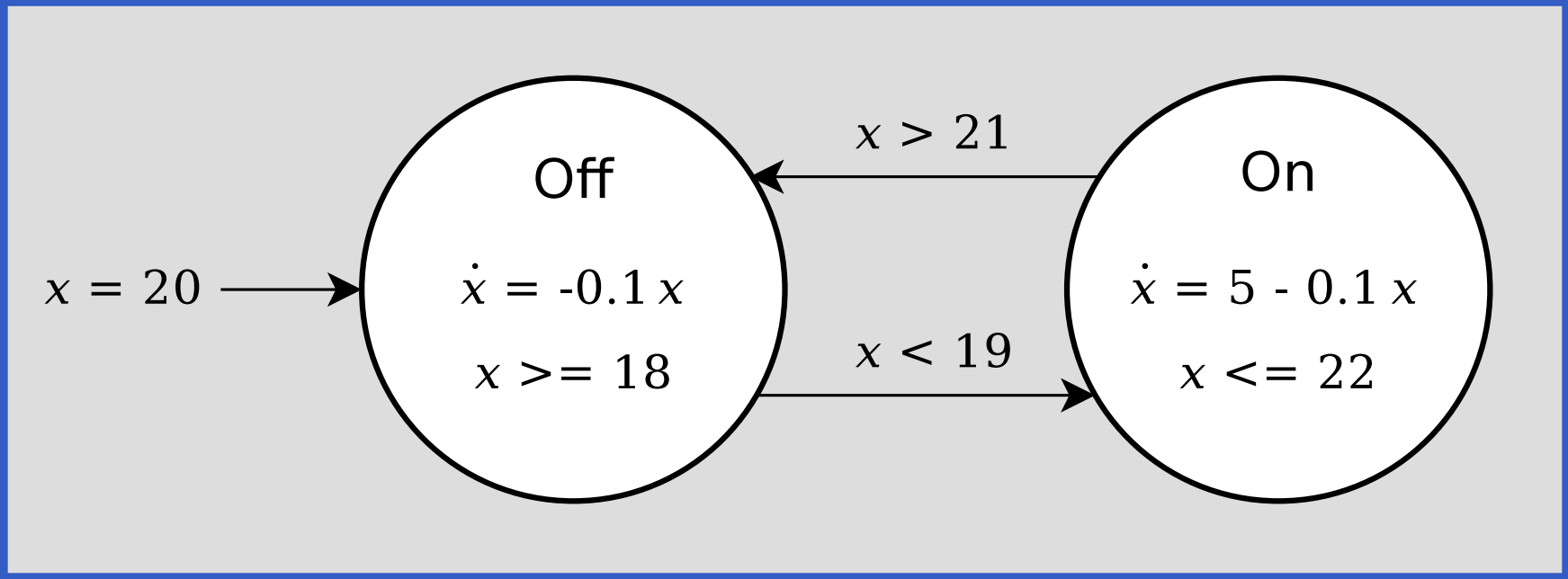
- ***Events***: finite set Σ of events and an edge labelling function $E \rightarrow \Sigma$ assigning event to each control switch $e \in E$.

Hybrid Automata (3)

- ***Events***: finite set Σ of events and an edge labelling function $E \rightarrow \Sigma$ assigning event to each control switch $e \in E$.

Note: Hybrid Automata arguably unrealistically expressive as events can be enforced at specific real-valued points in time. “Robust” or “Fuzzy” Hybrid Automata address this, but theory said to not differ significantly.

Thermostat Hybrid Automaton



Hybrid Automata Semantics (1)

Idea:

- States $Q, Q^0 \subseteq V \times \mathbb{R}^n$ such that invariants and, for Q^0, initial conditions satisfied.

Hybrid Automata Semantics (1)

Idea:

- States $Q, Q^0 \subseteq V \times \mathbb{R}^n$ such that invariants and, for Q^0, initial conditions satisfied.
- Discrete transitions $(v, \mathbf{x}) \xrightarrow{\sigma} (v', \mathbf{x}')$ iff control switch e from v to v', $\mathrm{jump}(e)[X, X := \mathbf{x}, \mathbf{x}']$, and $\mathrm{event}(e) = \sigma$.

Hybrid Automata Semantics (1)

Idea:

- States $Q, Q^0 \subseteq V \times \mathbb{R}^n$ such that invariants and, for Q^0, initial conditions satisfied.
- Discrete transitions $(v, \mathbf{x}) \xrightarrow{\sigma} (v', \mathbf{x}')$ iff control switch e from v to v', $\mathrm{jump}(e)[X, X := \mathbf{x}, \mathbf{x}']$, and $\mathrm{event}(e) = \sigma$.

Note: Typically infinite state space.

Hybrid Automata Semantics (2)

- For $\delta \in \mathbb{R}_{\geq 0}$, continuous transitions $(v, \mathbf{x}) \xrightarrow{\delta} (v, \mathbf{x}')$ iff there exists a differentiable function $f : [0, \delta] \to \mathbb{R}^n$ with first derivative $\dot{f}$ such that $f(0) = \mathbf{x}$, $f(\delta) = \mathbf{x}'$, and invariants and flow condititions satisfied for $f(\epsilon)$ and $\dot{f}(\epsilon)$ for all $\epsilon \in (0, \delta)$.

Hybrid Automata Semantics (2)

- For $\delta \in \mathbb{R}_{\geq 0}$, continuous transitions $(v, \mathbf{x}) \xrightarrow{\delta} (v, \mathbf{x}')$ iff there exists a differentiable function $f : [0, \delta] \rightarrow \mathbb{R}^n$ with first derivative $\dot{f}$ such that $f(0) = \mathbf{x}$, $f(\delta) = \mathbf{x}'$, and invariants and flow condititions satisfied for $f(\epsilon)$ and $\dot{f}(\epsilon)$ for all $\epsilon \in (0, \delta)$.

Note: Transition relation is generally highly non-deterministic.

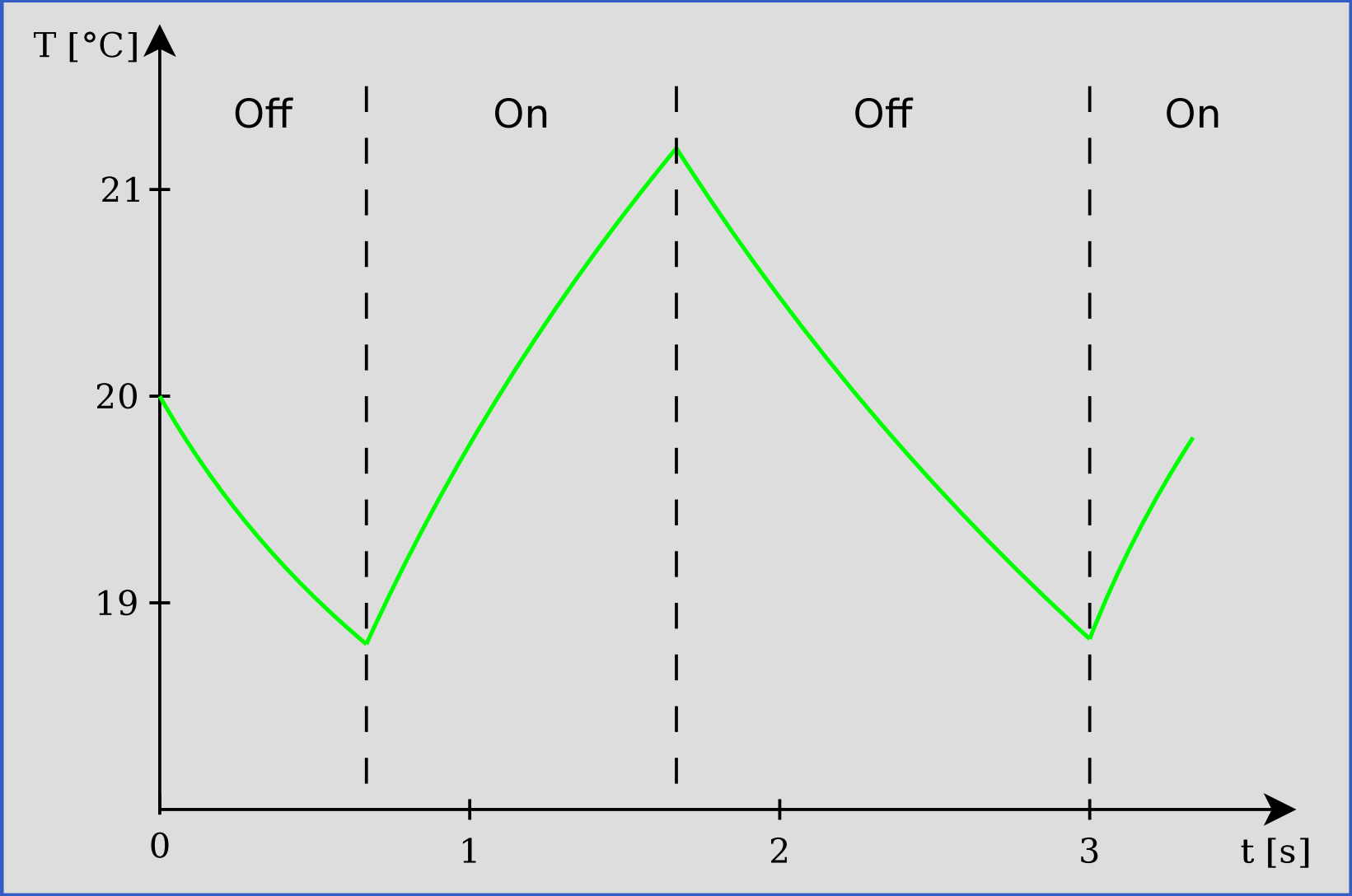
Hybrid Automata Semantics (2)

- For $\delta \in \mathbb{R}_{\geq 0}$, continuous transitions $(v, \mathbf{x}) \xrightarrow{\delta} (v, \mathbf{x}')$ iff there exists a differentiable function $f : [0, \delta] \to \mathbb{R}^n$ with first derivative $\dot{f}$ such that $f(0) = \mathbf{x}$, $f(\delta) = \mathbf{x}'$, and invariants and flow condititions satisfied for $f(\epsilon)$ and $\dot{f}(\epsilon)$ for all $\epsilon \in (0, \delta)$.

Note: Transition relation is generally highly non-deterministic.

Note: Additional liveness assumption: divergent time; i.e. there must exist sequences of transitions such that the sum of the labels goes to infinity.

Thermostat Behaviour



FHM in a Nutshell (1)

- ***Functional Hybrid Modelling (FHM)***:
  A functional approach to domain-specific languages for modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.

FHM in a Nutshell (1)

- ***Functional Hybrid Modelling (FHM)***:
  A functional approach to domain-specific languages for modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.
- Undirected equations: ***non-causal modelling***. (Differential Algebraic Equations, DAE)

FHM in a Nutshell (1)

- ***Functional Hybrid Modelling (FHM)***:
  A functional approach to domain-specific languages for modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.
- Undirected equations: ***non-causal modelling***. (Differential Algebraic Equations, DAE)
- Two-level design:
  - equation level for modelling components
  - functional level for spatial and temporal composition of components

FHM in a Nutshell (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.

FHM in a Nutshell (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.
- Spatial composition: signal relation ***application***; enables modular, hierarchical, system description.

FHM in a Nutshell (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.
- Spatial composition: signal relation ***application***; enables modular, hierarchical, system description.
- Temporal composition: ***switching*** from one structural configuration or control mode into another.

Hybrid Automata vs. FHM

FHM thus differs from Hybrid Automata in two central ways:

Hybrid Automata vs. FHM

FHM thus differs from Hybrid Automata in two central ways:

- Modular, hierarchical way to describe the system.

Hybrid Automata vs. FHM

FHM thus differs from Hybrid Automata in two central ways:

- Modular, hierarchical way to describe the system.
- ***A priori unbounded structural dynamism***: the next control mode ***computed*** as part of a discrete transition.

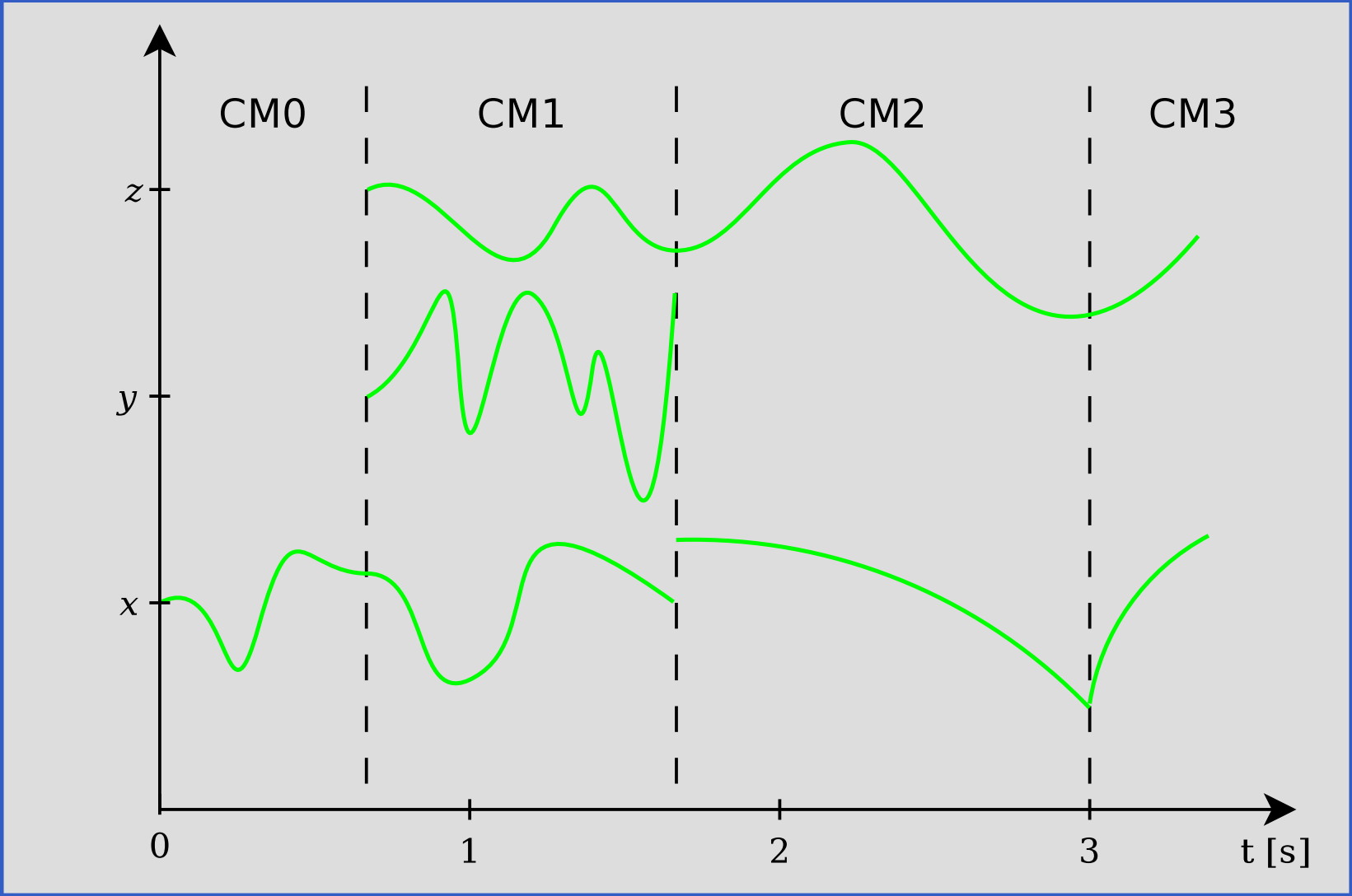
Hybrid Automata vs. FHM

FHM thus differs from Hybrid Automata in two central ways:

- Modular, hierarchical way to describe the system.
- ***A priori unbounded structural dynamism***: the next control mode ***computed*** as part of a discrete transition.

The latter enables modelling of “highly” structurally dynamic systems: systems where the number of structural configurations or modes is too large for an explicit enumeration to be practical or possible.

A Priori Unbounded Struct. Dynamism



FHM Semantics: Observations (1)

A complete semantics for FHM thus needs to account for:

(i) ***Flattening*** of hierarchical system description

(ii) ***Computation of next control mode*** given specific event and continuous state

(iii) ***Continuous behaviour*** of within each specific control mode

FHM Semantics: Observations (1)

A complete semantics for FHM thus needs to account for:

(i) ***Flattening*** of hierarchical system description

(ii) ***Computation of next control mode*** given specific event and continuous state

(iii) ***Continuous behaviour*** of within each specific control mode

(i) and (ii) are discrete aspects: we focus on those in the following.

FHM Semantics: Observations (2)

(iii) is essentially orthogonal to (i) and (ii): many possibilities depending on objectives:

- “Ideal” semantics: non executable specification, exact reals?
- Adopt approach similar to Hybrid Automata
- “Simulation” semantics: parametrised on specific solvers?
- Attempting to formally relate suitable ideal and simulation semantics? (Cf. Wan and Hudak: FRP from First Principles).

Causal vs. Non-Causal Modelling (1)

Causal or ***block-oriented*** modelling: model is ODE in ***explicit*** form:

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

Causality, i.e. cause-effect relationship, given by the modeller. Cf. Functional Programming.

Causal modelling is the dominating modelling paradigm; languages include Simulink.

Causal vs. Non-Causal Modelling (2)

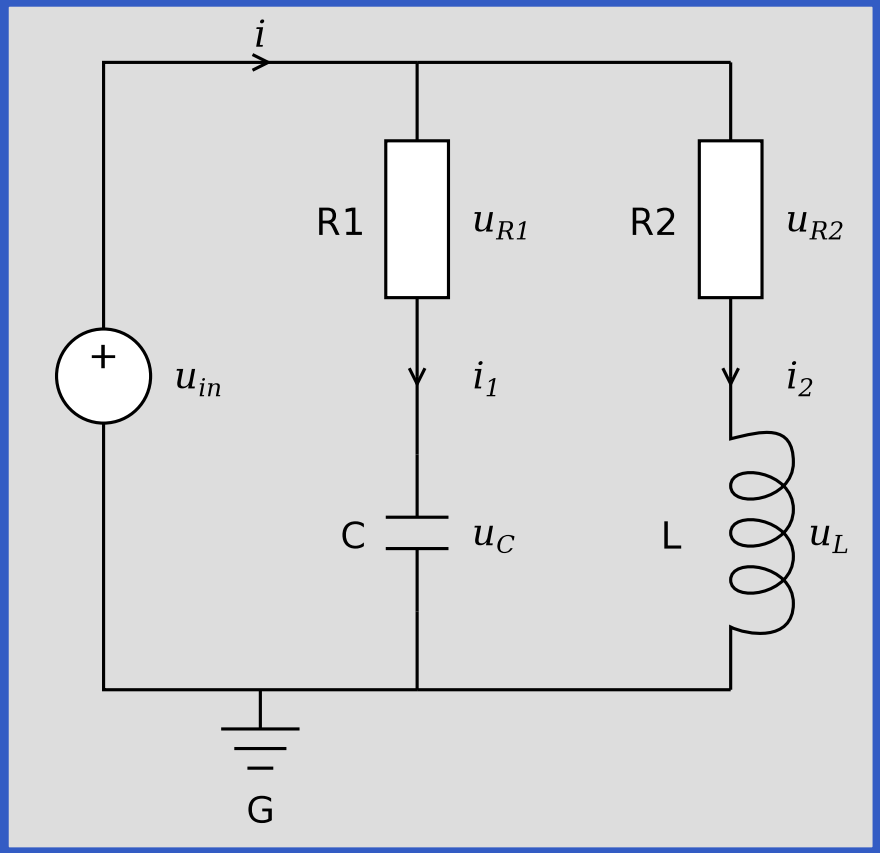
Non-causal or "***object-oriented***" modelling: model is DAE in implicit form:

$$\mathbf{f}(\mathbf{x}, \mathbf{x}', \mathbf{w}, \mathbf{u}, t) = \mathbf{0}$$

Causality inferred by simulation tool from usage context. Cf. Logic Programming, Constraint LP.

Non-causal modelling is a fairly recent development; languages include Dymola and Modelica.

Causal vs. Non-Causal Modelling (3)



$$
\begin{aligned}
u_{R_2} &= R_2 i_2 \\
u_L &= u_{in} - u_{R_2} \\
{i_2}' &= \frac{u_L}{L} \\
u_{R_1} &= u_{in} - u_C \\
i_1 &= \frac{u_{R_1}}{R_1} \\
{u_C}' &= \frac{i_1}{C} \\
i &= i_1 + i_2
\end{aligned}
$$

Non-Causal Modelling: Example (1)

Non-causal resistor model:

$$\begin{aligned} u &= v_p - v_n \\ i_p + i_n &= 0 \\ u &= R i_p \end{aligned}$$

Non-causal inductor model:

$$\begin{aligned} u &= v_p - v_n \\ i_p + i_n &= 0 \\ u &= L {i_p}' \end{aligned}$$

Non-Causal Modelling: Example (2)

Non-causal capacitor model:

$$
\begin{aligned}
u &= v_p - v_n \\
i_p + i_n &= 0 \\
i_p &= Cu'
\end{aligned}
$$

Note the commonality between the definitions; this can be factored out as a separate, abstract, ***two pin*** component.

Non-Causal Modelling: Example (3)

A non-causal model of the entire circuit is created by ***instantiating*** the component models: copy the equations and rename the variables.

The instantiated components are then ***composed*** by adding connection equations according to Kirchhoff's laws, e.g.:

$$\begin{aligned} v_{R_1,n} &= v_{C,p} \\ i_{R_1,n} + i_{C,p} &= 0 \end{aligned}$$

Very direct: can be accomplished through a drag-and-drop GUI.

Simple Circuit in FHM (1)

The type `Pin` is assumed to be a record type describing an electrical connection. It has fields v for voltage and i for current.

$$twoPin :: \texttt{SR}\ (\texttt{Pin}, \texttt{Pin}, \texttt{Voltage})$$
$$twoPin = \textbf{sigrel}\ (p, n, u)\ \textbf{where}$$
$$u = p.v - n.v$$
$$p.i + n.i = 0$$

Simple Circuit in FHM (2)

$$
\begin{aligned}
&\mathit{resistor} :: \texttt{Resistance} \rightarrow \texttt{SR}\,(\texttt{Pin}, \texttt{Pin}) \\
&\mathit{resistor}\ r = \textbf{sigrel}\ (p, n)\ \textbf{where} \\
&\qquad\qquad \mathit{twoPin} \diamond (p, n, u) \\
&\qquad\qquad r \cdot p.i = u \\
&\mathit{inductor} :: \texttt{Inductance} \rightarrow \texttt{SR}\,(\texttt{Pin}, \texttt{Pin}) \\
&\mathit{inductor}\ l = \textbf{sigrel}\ (p, n)\ \textbf{where} \\
&\qquad\qquad u1 = p.v - n.v \\
&\qquad\qquad p.i + n.i = 0 \\
&\qquad\qquad l \cdot \textbf{der}(p.i) = u
\end{aligned}
$$

Simple Circuit in FHM (3)

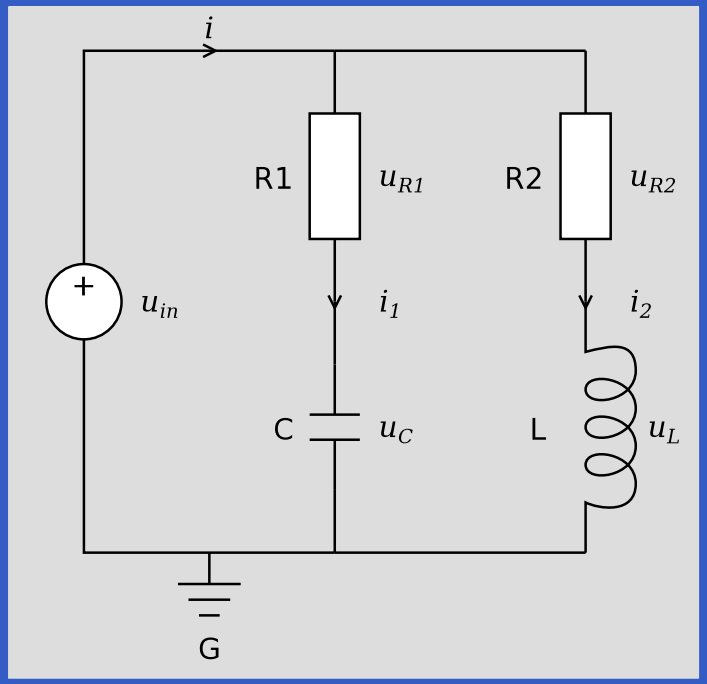
Example of signal relation application:

$$resistor\ 2200 = \textbf{sigrel}\ (p, n)\ \textbf{where}$$
$$u1 = p.v - n.v$$
$$p.i + n.i = 0$$
$$2200 \cdot p.i = u1$$

Simple Circuit in FHM (4)

$$
\begin{aligned}
&\mathit{simpleCircuit} :: \texttt{SR Current} \\
&\mathit{simpleCircuit} = \textbf{sigrel i where} \\
&\qquad \mathit{resistor}(1000) \diamond (\mathit{r1p}, \mathit{r1n}) \\
&\qquad \mathit{resistor}(2200) \diamond (\mathit{r2p}, \mathit{r2n}) \\
&\qquad \mathit{capacitor}(0.00047) \diamond (\mathit{cp}, \mathit{cn}) \\
&\qquad \mathit{inductor}(0.01) \diamond (\mathit{lp}, \mathit{ln}) \\
&\qquad \mathit{vSourceAC}(12) \diamond (\mathit{acp}, \mathit{acn}) \\
&\qquad \mathit{ground} \diamond \mathit{gp} \\
&\qquad \ldots
\end{aligned}
$$

Simple Circuit in FHM (5)



...

connect acp, $r1p$, $r2p$

connect $r1n$, cp

connect $r2n$, lp

connect acn, cn, ln, gp

$i = r1p.i + r2p.i$

Structural Dynamism: Ideal Diode

$icDiode$ :: `SR` (`Pin`, `Pin`)

$icDiode =$ **sigrel** (p, n) **where**

$twoPin \diamond (p, n, u)$

initially; **when** $p.v - n.v > 0 \;\Rightarrow$

$u = 0$

when $p.i < 0 \;\Rightarrow$

$p.i = 0$

FHM Semantics: Bird's-eye View (1)

Structure of the discrete part of the semantics:

$$
\begin{aligned}
flatten &: FHMModel \rightarrow StaticState \\
extract &: StaticState \\
&\rightarrow (FlatEqs, FlatEqs, Conds) \\
processEvents &: StaticState \rightarrow Events \\
&\rightarrow StaticState
\end{aligned}
$$

Thus, "co-inductive", at least in spirit.

FHM Semantics: Bird's-eye View (2)

- In current semantic formalisation, $FHMModel$ and $StaticState$ indexed on top-level signal-relation type.
- We would like to index $StaticState$ also on:
  - Accepted events
  - Type of accepted ***dynamic*** state

  That way, we'd have a precise handle on the communication protocol between the discrete and the continuous part of the semantics.

Prototype Hydra Implementation (1)

The current FHM instance is called ***Hydra***:

- Embedding in ***Haskell***.
- Model ***transformed*** to form suitable for simulation, then ***JIT compiled to native code*** by an embedded compiler.
- State-of-the art ***numerical solvers from SUNDIALS*** suite (from LLNL) used for simulation and event detection.
- Transformation and compilation ***repeated*** when system structure changes at events.

Prototype Hydra Implementation (2)