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: Standard approach for semantics of hybrid systems:

Hybrid Automata (1)

Hybrid Automata: Standard approach for semantics of hybrid systems:


- **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 \(X, X, \text{ and } X \cup \dot{X}\) 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 \(e \in E\)
Events: finite set $\Sigma$ 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 $\Sigma$ 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

\[ x = 20 \]

```
Off
\begin{align*}
\dot{x} &= -0.1 \cdot x \\
x &\geq 18
\end{align*}
```

```
On
\begin{align*}
\dot{x} &= 5 - 0.1 \cdot x \\
x &\leq 22
\end{align*}
```

\[ x > 21 \]

\[ x < 19 \]
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 $(v, x) \xrightarrow{\sigma} (v', x')$ iff control switch $e$ from $v$ to $v'$, $\text{jump}(e)[X, X := x, x']$, and $\text{event}(e) = \sigma$. 
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 $(v, x) \xrightarrow{\sigma} (v', x')$ iff control switch $e$ from $v$ to $v'$, $\text{jump}(e)[X, X := x, x']$, and $\text{event}(e) = \sigma$.

**Note**: Typically infinite state space.
Hybrid Automata Semantics (2)

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

**Note**: Transition relation is generally highly non-deterministic.
For $\delta \in \mathbb{R}_{\geq 0}$, continuous transitions $(v, x) \xrightarrow{\delta} (v, x')$ iff there exists a differentiable function $f : [0, \delta] \rightarrow \mathbb{R}^n$ with first derivative $\dot{f}$ such that $f(0) = x$, $f(\delta) = x'$, and invariants and flow conditions 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

![Graph showing thermostat behaviour over time](image)
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)
**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
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.
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 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
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.
(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:

\[ x' = f(x, u, t) \]

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

Causal modelling is the dominating modelling paradigm; languages include Simulink.
Non-causal or “object-oriented” modelling: model is DAE in implicit form:

\[ f(x, x', w, u, t) = 0 \]

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)

\[
\begin{align*}
    u_{R2} &= R_2 i_2 \\
    u_L &= u_{\text{in}} - u_{R2} \\
    \dot{i}_2' &= \frac{u_L}{L} \\
    u_{R1} &= u_{\text{in}} - u_C \\
    \dot{i}_1 &= \frac{u_{R1}}{R_1} \\
    u_C' &= \frac{i_1}{C} \\
    \dot{i} &= i_1 + i_2
\end{align*}
\]
Non-Causal Modelling: Example (1)

Non-causal resistor model:

\[ u = v_p - v_n \]
\[ i_p + i_n = 0 \]
\[ u = Ri_p \]

Non-causal inductor model:

\[ u = v_p - v_n \]
\[ i_p + i_n = 0 \]
\[ u = Li_p' \]
Non-Causal Modelling: Example (2)

Non-causal capacitor model:

\[
\begin{align*}
    u &= v_p - v_n \\
    i_p + i_n &= 0 \\
    i_p &= Cu'
\end{align*}
\]

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

\[
\begin{align*}
    v_{R_1,n} &= v_{C,p} \\
    i_{R_1,n} + i_{C,p} &= 0
\end{align*}
\]

Very direct: can be accomplished through a drag-and-drop GUI.
The type \texttt{Pin} is assumed to be a record type describing an electrical connection. It has fields $v$ for voltage and $i$ for current.

\begin{align*}
\text{twoPin} \Colon= \text{SR}(\text{Pin}, \text{Pin}, \text{Voltage}) \\
\text{twoPin} = \text{sigrel}(p, n, u) \quad \text{where} \\
& u = p.v - n.v \\
& p.i + n.i = 0
\end{align*}
Simple Circuit in FHM (2)

resistor :: Resistance → SR(Pin, Pin)
resistor \( r = \text{sigrel}(p, n) \) where
\[
twoPin \diamond (p, n, u) \\
r \cdot p.\!i = u
\]

inductor :: Inductance → SR(Pin, Pin)
inductor \( l = \text{sigrel}(p, n) \) where
\[
\begin{align*}
u_1 &= p.v - n.v \\
p.\!i + n.\!i &= 0 \\
l \cdot \text{der}(p.\!i) &= u
\end{align*}
\]
Example of signal relation application:

\[ \text{resistor 2200} = \text{sigrel}(p, n) \text{ where} \]
\[ u_1 = p.v - n.v \]
\[ p.i + n.i = 0 \]
\[ 2200 \cdot p.i = u_1 \]
simpleCircuit :: SR Current

simpleCircuit = sigrel i where

resistor(1000) ◦ (r1p, r1n)
resistor(2200) ◦ (r2p, r2n)
capacitor(0.00047) ◦ (cp, cn)
inductor(0.01) ◦ (lp, ln)
vSourceAC(12) ◦ (acp, acn)
ground ◦ gp

...
Simple Circuit in FHM (5)

\[ i = r_1 \cdot i + r_2 \cdot i \]

**connect** acp, r1p, r2p
**connect** r1n, cp
**connect** r2n, lp
**connect** acn, cn, ln, gp

Towards a Formal Semantics for FHM, Part I – p.26/31
Structural Dynamism: Ideal Diode

\textit{icDiode} :: \texttt{SR(\textit{Pin}, \textit{Pin})}
\textit{icDiode} = \texttt{sigrel}(p, n) \textbf{where}
\texttt{twoPin} \diamond (p, n, u)
\textbf{initially; when} \ p.v - n.v > 0 \Rightarrow
\hspace{1cm} u = 0
\textbf{when} \ p.i < 0 \Rightarrow
\hspace{1cm} p.i = 0
Structure of the discrete part of the semantics:

\[
\begin{align*}
\text{flatten} &: \quad \text{FHMModel} \to \text{StaticState} \\
\text{extract} &: \quad \text{StaticState} \\
&\quad \to (\text{FlatEqs}, \text{FlatEqs}, \text{Conds}) \\
\text{processEvents} &: \quad \text{StaticState} \to \text{Events} \\
&\quad \to \text{StaticState}
\end{align*}
\]

Thus, “co-inductive”, at least in spirit.
In current semantic formalisation, \textit{FHMModel} and \textit{StaticState} indexed on top-level signal-relation type.

We would like to index \textit{StaticState} also on:

- Accepted events

- Type of accepted \textit{dynamic} state

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