Towards a UTP semantics for Modelica

Simon Foster Bernhard Thiele
Ana Cavalcanti Jim Woodcock

4th June 2016

Horizon 2020 Programme

Table of Contents

Introduction

Hybrid Relations

Semantics of Modelica

Future work

Table of Contents

Introduction

Hybrid Relations

Semantics of Modelica

Future work

INTO-CPS project

- integrated tool-chain for MBD of Cyber-Physical Systems
- “multi-modelling” with heterogeneous models and languages
- SysML for high-level system modelling
- VDM-RT for modelling discrete controllers
- Modelica and 20-sim for continuous system dynamics
- FMI for multi-modelling of heterogeneous systems
- four industrial case studies
  - route simulation in electric cars (TWT)
  - agricultural robot (Agro Intelligence)
  - railways (ClearSy)
  - HVACs in smart buildings (UTRC)
- semantic integration using Unifying Theories of Programming

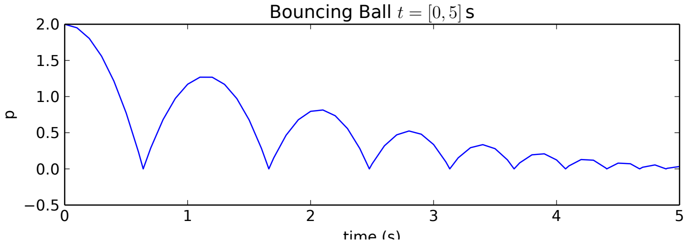
Modelica language

- industrial language for modelling hybrid dynamical systems
- provides language of continuous blocks with connections
- based on hybrid differential-algebraic equations
- combines DAEs with an event handling mechanism
- triggers based on continuous variable conditions
- can cause discontinuities in evolution
- implementations include:
  - Dymola
  - Wolfram SystemModeler
  - MapleSim
  - OpenModelica (INTO-CPS partner)
- incomplete formal semantics

Example 1: Bouncing Ball

Bouncing ball in Modelica

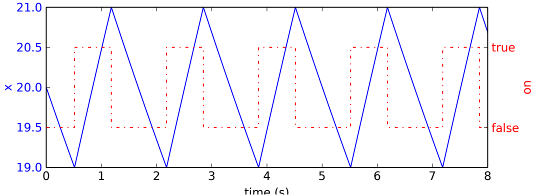
```
model BouncingBall
  Real p(start=2,fixed=true), v(start=0,fixed=true);
equation
  der(v) = -9.81;
  der(p) = v;
  when p <= 0 then
    reinit(v, -0.8*v);
  end when;
end BouncingBall;
```



Example 2: Thermostat

Thermostat in Modelica

```
model Thermostat
  Real x(start=20,fixed=true);
  Boolean on(start=false,fixed=true);
equation
  when x < 19 then
    on = true;
  elsewhen x > 21 then
    on = false;
  end when;
  der(x) = if on then 5 - 0.1*x else -0.1*x;
end Thermostat;
```



UTP in brief

- alphabetised relational calculus – everything is a relation
- expressed as predicates over input, output variables (x / x')
- predicates encode the set of observable behaviours

UTP in brief

- alphabetised relational calculus – everything is a relation
- expressed as predicates over input, output variables (x / x')
- predicates encode the set of observable behaviours

$$
\begin{aligned}
x := v &\triangleq x' = v \wedge y' = y \\
P \,;\, Q &\triangleq \exists\, x_0 \bullet P[x_0/x'] \wedge Q[x_0/x] \\
P \lhd b \rhd Q &\triangleq (b \wedge P) \vee (\neg b \wedge Q) \\
P^{*} &\triangleq \nu X \bullet P \,;\, X
\end{aligned}
$$

UTP in brief

- alphabetised relational calculus – everything is a relation
- expressed as predicates over input, output variables (x / x')
- predicates encode the set of observable behaviours

$$
\begin{aligned}
x := v &\triangleq x' = v \wedge y' = y \\
P \,;\, Q &\triangleq \exists\, x_0 \bullet P[x_0/x'] \wedge Q[x_0/x] \\
P \lhd b \rhd Q &\triangleq (b \wedge P) \vee (\neg b \wedge Q) \\
P^* &\triangleq \nu X \bullet P \,;\, X
\end{aligned}
$$

- how to go beyond simple imperative behaviour?
- UTP theories to isolate paradigmatic aspects of a language
- compose theories to produce heterogeneous semantic models
- e.g. object-orientation, concurrency, real-time, ODEs

UTP theories

1. observational variables
   - different from program variables: encode model properties
   - e.g. $ti, ti' : \mathbb{R}$ to represent start/end time

UTP theories

1. observational variables
   - different from program variables: encode model properties
   - e.g. $ti, ti' : \mathbb{R}$ to represent start/end time
2. healthiness conditions
   - constrain the behaviour of the observational variables
   - e.g. $ti' \geq ti$ – time moves forward
   - often expressed as idempotent, monotone functions over preds
   - ensures that healthy predicates form a complete lattice

UTP theories

1. observational variables
   - different from program variables: encode model properties
   - e.g. $ti, ti' : \mathbb{R}$ to represent start/end time
2. healthiness conditions
   - constrain the behaviour of the observational variables
   - e.g. $ti' \geq ti$ – time moves forward
   - often expressed as idempotent, monotone functions over preds
   - ensures that healthy predicates form a complete lattice
3. signature
   - the operators of the language
   - e.g. $\mathsf{Wait}\, n \triangleq ti' = ti + n \wedge v' = v$
   - closed under the healthiness conditions (well-behaved)

Approach

1. create a hybrid relational calculus
   - extends alphabetised relational calculus
   - inspired by Hybrid CSP and Duration Calculus
   - combine continuous variables and discrete i/o variables
   - imperative operators from relational calculus
   - differential equations and pre-emption
   - mechanised in Isabelle/UTP proof assistant

Approach

1. create a hybrid relational calculus
   - extends alphabetised relational calculus
   - inspired by Hybrid CSP and Duration Calculus
   - combine continuous variables and discrete i/o variables
   - imperative operators from relational calculus
   - differential equations and pre-emption
   - mechanised in Isabelle/UTP proof assistant
2. define semantic mapping from Modelica to hybrid relations
   - currently a direct semantics for flattened hybrid DAEs
   - describe evolution of ODEs and DAEs
   - elaborate event handling mechanism

Table of Contents

Introduction

Hybrid Relations

Semantics of Modelica

Future work

Motivation

- augment relational calculus with continuous variables
- support modelling using differential equations $\langle \dot{x} = \mathcal{F}(x, \dot{x}) \rangle$
- retain standard discrete operators defs – e.g. $P \,;\, Q$, $x := v$
- characterise continuous behaviour with healthiness conditions
- combine with other theories, e.g. CSP and reactive designs
- inspirations:
  - Hybrid CSP (He, Zhan et al.) – DAEs and pre-emption
  - HRML (He) – tri-partite alphabet
  - Duration Calculus (Zhu et al.) – interval operator
  - Timed Reactive Designs (Hayes et al.)

Hybrid relational calculus

- formalise key operators of hybrid behaviour for Modelica
- we begin with a kernel language of imperative hybrid programs
- operators given a semantics in the theory of hybrid relations
- discrete relational operators
  - sequential composition — $P \mathbin{;} Q$
  - assignment — $x := v$
  - if-then-else conditional — $P \lhd b \rhd Q$
  - iteration — P^* and P^ω
- continuous evolution operators
  - DAE — $\langle\, \underline{\dot{v}}_1 = f_1;\ \cdots;\ \underline{\dot{v}}_n = f_n \mid B \,\rangle$
  - pre-emption — $P \,[\, B \,]\, Q$
  - interval (continuous invariant) — $\llbracket P \rrbracket$

Example 1: Simple Bouncing Ball

Bouncing ball in Modelica

```
model BouncingBall
  Real p(start=2,fixed=true), v(start=0,fixed=true);
equation
  der(v) = -9.81;
  der(p) = v;
  when p <= 0 then
    reinit(v, -0.8*v);
  end when;
end BouncingBall;
```

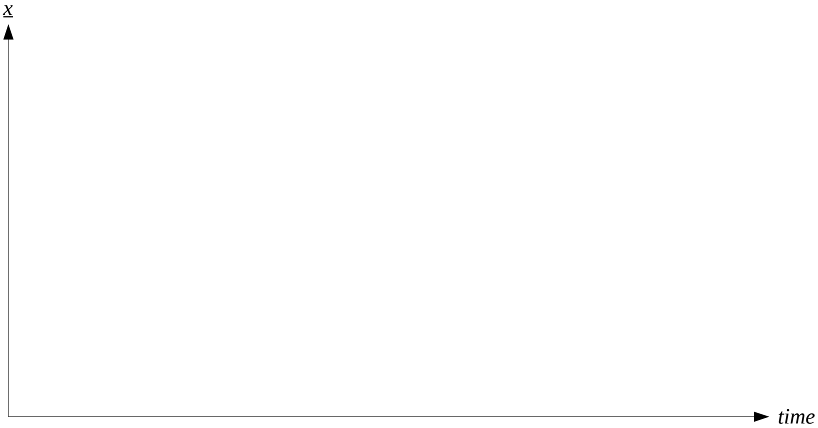
Bouncing ball in hybrid relational calculus

$$p, v := 2, 0 \; ; \left(\left\langle \, \dot{\underline{p}} = \underline{v}; \; \dot{\underline{v}} = -9.81 \, \right\rangle \left[\, \underline{p} \leq 0 \, \right] v := -v * .8 \right)^{\omega}$$

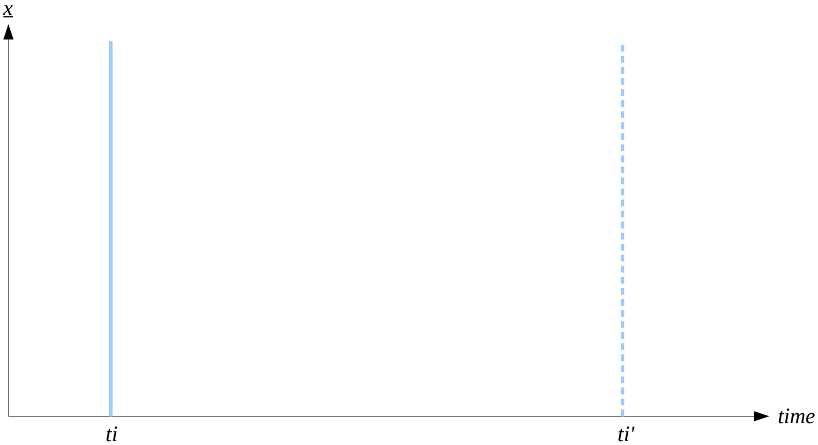
UTP theory of hybrid relations

- model for hybrid relational calculus
- $\alpha(P) = \mathrm{in}\alpha(P) \cup \mathrm{out}\alpha(P) \cup \mathrm{con}\alpha(P)$
- $x, x' : T$ discrete input and output variables
- $\underline{x} : \mathbb{R}_{\geq 0} \rightarrow T$ total continuous variable trajectories
- $ti, ti' : \mathbb{R}_{\geq 0}$ record beginning and end of current observation
- continuous variables range over right-open interval $[ti, ti')$
- accompanied by discrete copies at ends of interval
- continuous linking invariants: $x = \underline{x}(ti)$ and $x' = \lim_{t \to ti'} (\underline{x}(t))$

Healthy continuous observations

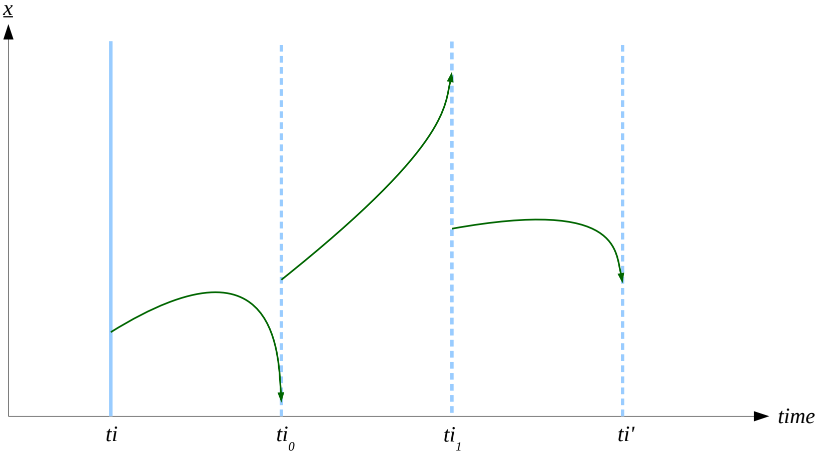


HCT1: Interval has start and end



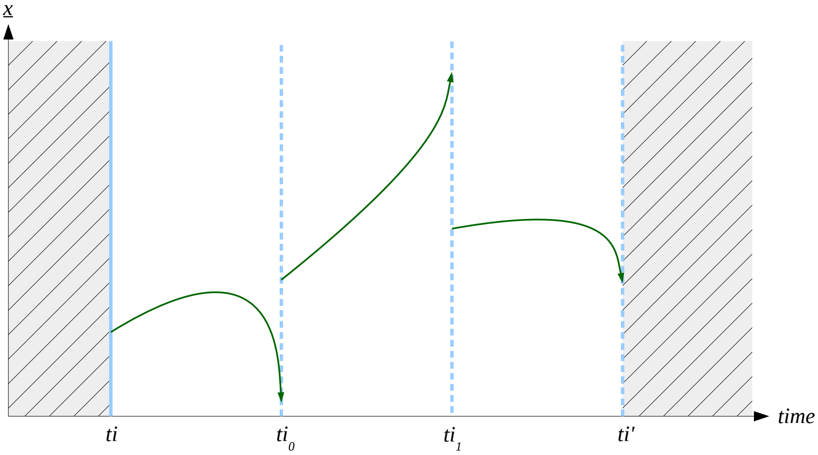
$$\textbf{\textit{HCT1}}(P) \triangleq P \wedge ti \leq ti'$$

HCT2: Trajectories are piecewise continuous



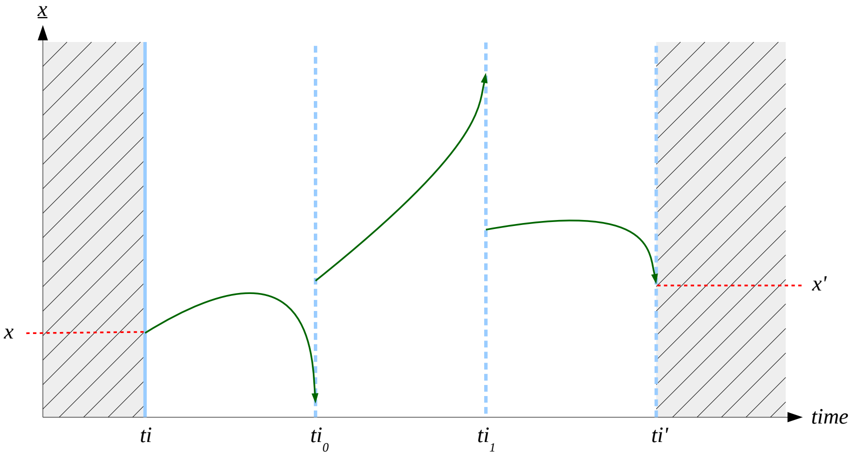
$$\textbf{\textit{HCT2}}(P) \triangleq P \wedge \left(ti < ti' \Rightarrow \left(\begin{array}{l} \exists\, I : \mathbb{R}_{\mathsf{oseq}} \bullet \operatorname{ran}(I) \subseteq \{ti \ldots ti'\} \\ \wedge \{ti, ti'\} \subseteq \operatorname{ran}(I) \wedge \\ \wedge (\forall\, n < \#I - 1 \bullet \underline{x} \ \mathsf{cont\text{-}on}\, [I_n, I_{n+1}]) \end{array} \right) \right)$$

HCT3: Observations are past and future independent



$$\textbf{\textit{HCT3}}(P) \triangleq \bigsqcap t \notin [ti, ti'), v \in \mathsf{type}(\underline{x}) \bullet (P \wedge \underline{x}(t) = v)$$

Continuous linking invariants



$x = \underline{x}(ti)$ and $x' = \lim_{t \to ti'} (\underline{x}(t))$

Healthiness conditions summary

$$
\begin{aligned}
\textbf{\textit{HCT1}}(P) &\triangleq P \land ti \leq ti' \\
\textbf{\textit{HCT2}}(P) &\triangleq P \land \left(ti < ti' \Rightarrow \left(\begin{array}{l} \exists\, I : \mathbb{R}_{\mathsf{oseq}} \bullet \mathrm{ran}(I) \subseteq \{ti \ldots ti'\} \\ \quad \land \{ti, ti'\} \subseteq \mathrm{ran}(I) \land \\ \quad \land (\forall\, n < \#I - 1 \bullet \\ \qquad\qquad \underline{x} \ \textsf{cont-on}\, [I_n, I_{n+1})) \end{array} \right) \right) \\
\textbf{\textit{HCT3}}(P) &\triangleq \bigsqcap t \notin [ti, ti'), v \in \textsf{type}(\underline{x}) \bullet (P \land \underline{x}(t) = v) \\
\textbf{\textit{HCT}}(P) &\triangleq \textbf{\textit{HCT1}} \circ \textbf{\textit{HCT2}} \circ \textbf{\textit{HCT3}}(P)
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbb{R}_{\mathsf{oseq}} &\triangleq \{x : \textsf{seq}\, \mathbb{R} \mid \forall\, n < \#x - 1 \bullet x_n < x_{n+1}\} \\
f \ \textsf{cont-on}\, [m, n) &\triangleq \forall\, t \in [m, n) \bullet \lim_{x \to t} f(x) = f(t)
\end{aligned}
$$

Behaviours

- **instantaneous behaviour**
  - instantaneous observations – $ti' = ti$
  - do not contribute to trajectories ($[ti, ti) = \emptyset$)
  - assign values only to discrete variables
  - described by imperative and concurrent programming operators
  - sequential behaviours $P \,;\, Q$ occupy instant ti
- **continuous time behaviour**
  - non-zero observation duration – $ti' > ti$
  - evolution described by piecewise continuous functions
  - discrete copies of continuous variables at beginning and end
  - described by systems of ODEs and DAEs
  - no sharing of instants between $P \,;\, Q$

Discrete denotational semantics

- standard definitions

$$
\begin{aligned}
x := v &\triangleq x' = v \wedge y' = y \\
P \; ; \; Q &\triangleq \exists\, x_0 \bullet P[x_0/x'] \wedge Q[x_0/x] \\
P \lhd b \rhd Q &\triangleq (b \wedge P) \vee (\neg b \wedge Q) \\
P^{*} &\triangleq \nu X \bullet P \; ; \; X
\end{aligned}
$$

- sequential composition takes the trajectory conjunctions

Continuous denotational semantics

$$\lceil P \rceil \triangleq \textbf{\textit{HCT2}}(\ell > 0 \wedge (\forall \underline{t} \in [ti, ti') \bullet P \,@\, \underline{t}))$$

$$[\![P]\!] \triangleq \lceil P \rceil \wedge \bigwedge_{\underline{v} \in \mathrm{con}\alpha(P)} (v = \underline{v}(ti) \wedge v' = \lim_{t \to ti'} (\underline{v}(t))) \wedge I\!I_{\mathrm{dis}\alpha(P)}$$

$$\langle\, \dot{\underline{v}}_1 = f_1;\ \cdots;\ \dot{\underline{v}}_n = f_n \,|\, B \,\rangle \triangleq$$
$$[\![(\forall i \in 1..n \bullet \dot{\underline{v}}_i(\underline{t}) = f_i(\underline{t}, \underline{v}_1(\underline{t}), \cdots, \underline{v}_n(\underline{t}))) \wedge B]\!]$$

$$P\,[\,B\,]\,Q \triangleq (Q \lhd B \,@\, ti \rhd (P \wedge \lceil \neg B \rceil)) \vee ((\lceil \neg B \rceil \wedge B \,@\, ti' \wedge P)\,;\,Q)$$

- **Theorem:** all operators are ***HCT*-closed**

Mechanisation

- based on Isabelle/UTP and the Multivariate Analysis package
- mimics syntax contained in paper
- proof support for alphabetised and hybrid relational calculi
- real numbers based on Cauchy sequences
- support for limits, ODEs, and their solutions
- proved key properties of HCT and signature

Table of Contents

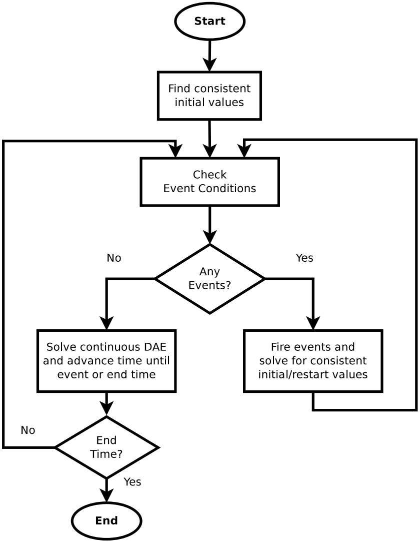
Introduction

Hybrid Relations

Semantics of Modelica

Future work

Event iteration cycle



Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica

Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica
- variables: dynamic (x), algebraic (y), discrete (q)

Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica

- variables: dynamic (x), algebraic (y), discrete (q)
- discrete variables only change at events

Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica

- variables: dynamic (x), algebraic (y), discrete (q)
- discrete variables only change at events
- $k \in \mathbb{N}_{>0}$ conditional DAEs:
  - differential equations $\dot{x} = \mathcal{F}_i(x, y, q)$ for $i \in 1..k$
  - algebraic equations $y = \mathcal{B}_i(x, y, q)$ for $i \in 1..k$
  - boolean DAE guards $\mathcal{G}_i(x, y, q)$ for $i \in 1..k-1$

Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica

- variables: dynamic (x), algebraic (y), discrete (q)
- discrete variables only change at events
- $k \in \mathbb{N}_{>0}$ conditional DAEs:
  - differential equations $\dot{x} = \mathcal{F}_i(x, y, q)$ for $i \in 1..k$
  - algebraic equations $y = \mathcal{B}_i(x, y, q)$ for $i \in 1..k$
  - boolean DAE guards $\mathcal{G}_i(x, y, q)$ for $i \in 1..k-1$
- $l \in \mathbb{N}$ boolean event conditions $\mathcal{C}_i(x, y, q)$ for $i \in 1..l$

Denotational semantics of Modelica

- currently defined as a mapping on flat Modelica
- variables: dynamic (x), algebraic (y), discrete (q)
- discrete variables only change at events
- $k \in \mathbb{N}_{>0}$ conditional DAEs:
  - differential equations $\dot{x} = \mathcal{F}_i(x, y, q)$ for $i \in 1..k$
  - algebraic equations $y = \mathcal{B}_i(x, y, q)$ for $i \in 1..k$
  - boolean DAE guards $\mathcal{G}_i(x, y, q)$ for $i \in 1..k-1$
- $l \in \mathbb{N}$ boolean event conditions $\mathcal{C}_i(x, y, q)$ for $i \in 1..l$
- $m \in \mathbb{N}$ conditional discrete equation blocks
  - discrete-event guards $\mathcal{H}_{i,j}(x, y, q, q_{pre})$ for $i \in 1..m$, $j \in 1..n$
  - discrete algorithms $\mathcal{P}_{i,j}(x, y, q, q_{pre})$ for $i \in 1..m$, $j \in 1..n$

High-level semantic mapping

$$
\begin{aligned}
\mathcal{M} &= \mathsf{Init}\ ;\ (\mathsf{DAE}\,[\,\mathsf{Events}\,]\,\mathsf{Discr})^{\omega} \\
\mathsf{Init} &= \underline{x}, \underline{y}, q := u, v, w \\
\mathsf{DAE} &= \left\langle\, \underline{x} = \mathcal{F}_1(\underline{\dot{x}}, \underline{y}, q) \,\middle|\, \mathcal{B}_1(\underline{x}, \underline{y}, q) \,\right\rangle \lhd \mathcal{G}_1 \rhd \cdots \\
& \quad \lhd \mathcal{G}_{n-1} \rhd \left\langle\, \underline{\dot{x}} = \mathcal{F}_n(\underline{x}, \underline{y}, q) \,\middle|\, \mathcal{B}_n(\underline{x}, \underline{y}, q) \,\right\rangle \\
\mathsf{Events} &= \bigvee_{i \in \{1..k\}} \mathcal{C}_i(\underline{x}, \underline{y}, q) \neq \mathcal{C}_i(x, y, q) \\
\mathsf{Discr} &= \textbf{\textit{var}}\ q_{pre} \bullet \\
& \quad \textbf{\textit{until}}\ q_{pre} = q\ \textbf{\textit{do}} \\
& \qquad q_{pre} := q\ ; \\
& \qquad \mathcal{P}_{1,1}(\underline{x}, \underline{y}, q, q_{pre}) \lhd \mathcal{H}_{1,1}(\underline{x}, \underline{y}, q, q_{pre}) \rhd \mathcal{P}_{1,2}(\underline{x}, \underline{y}, q) \lhd \cdots\ ;\ \cdots\ ; \\
& \qquad \mathcal{P}_{m,1}(\underline{x}, \underline{y}, q, q_{pre}) \lhd \mathcal{H}_{m,1}(\underline{x}, \underline{y}, q, q_{pre}) \rhd \mathcal{P}_{m,2}(\underline{x}, \underline{y}, q, q_{pre}) \lhd \cdots\ ; \\
& \quad \textbf{\textit{od}}
\end{aligned}
$$

Semantics of bouncing ball

Example

Bouncing ball semantics in hybrid relational calculus

$$
\begin{array}{l}
h, v, c := 2, 0, \mathit{false} \,; \\
\quad (\left\langle\, \underline{\dot{v}} = -9.81;\; \underline{\dot{h}} = \underline{v} \,\right\rangle \\
\qquad [(\underline{h} < 0) \neq (h < 0)] \\
\qquad \textbf{\textit{var}}\ c_{pre} \bullet \\
\qquad\quad \textbf{\textit{until}}\ (c_{pre} = c)\ \textbf{\textit{do}} \\
\qquad\qquad c_{pre} := c \,;\, c := h < 0 \,; \\
\qquad\qquad v := -0.8 \cdot v \lhd c \land \neg c_{pre} \rhd I\!I \\
\qquad\quad \textbf{\textit{od}})^{\omega}
\end{array}
$$

Table of Contents

Introduction

Hybrid Relations

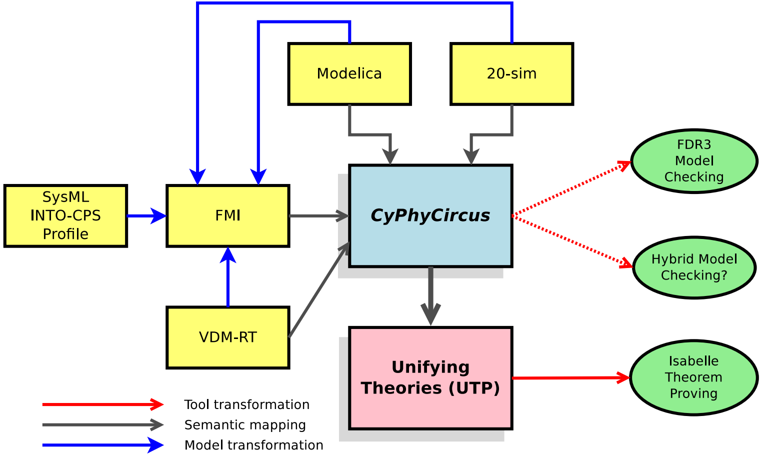
Semantics of Modelica

Future work

Future work

- completion of the Modelica semantic mapping
  - e.g. full expression AST
- compositional mapping for blocks and wires
- testing semantics on more substantive examples
- theorem proving support for Modelica
  - are my initial value constraints consistent?
  - does the event iteration cycle terminate?
  - does this algorithm satisfy an invariant?
- enrich hybrid relations $\longrightarrow$ hybrid reactive designs
- use the latter to build a lingua franca – CyPhyCircus
- enable integration of discrete controllers with continuous plant

INTO-CPS semantic integration



CyPhyCircus

- minimal CSP extension with hybrid behaviour
- semantics based on hybrid reactive designs $\boldsymbol{HR}(P \vdash Q)$
- enable modelling of reactive and concurrent hybrid systems
- give a model to Hybrid Hoare Logic using interval operator(?)

$$
\begin{aligned}
P, Q ::= \; & \mathsf{Skip} \mid \mathsf{Stop} \mid P \,;\, Q \mid P \lhd b \rhd Q \mid P \sqcap Q \mid x := e \\
& \mid P^{*} \mid P^{\omega} \mid \langle\, \underline{\dot{x}} = f(x, \underline{x}, \underline{\dot{x}}) \mid b(x, \underline{x}) \,\rangle \mid P \,[\, b(x, \underline{x}) \,]\, Q \\
& \mid \mathbin{[\!]}_{i \in I} \; a_i?x \rightarrow P(x) \mid a!e \rightarrow P \mid P \mathbin{\triangle} Q \mid P \parallel Q
\end{aligned}
$$

Links

- Isabelle/UTP git repository: https://github.com/isabelle-utp/utp-main/tree/shallow.2016
- INTO-CPS project: http://into-cps.au.dk/

Thanks for listening!

Horizon 2020 Programme