Towards a Mechanised Denotational Semantics for Modelica

Simon Foster† Bernhard Thiele‡
Jim Woodcock† Peter Fritzson‡

†Department of Computer Science, University of York
‡PELAB, Linköping University

3rd February 2016
INTO-CPS project

- aim: creation of an integrated tool-chain for model-based design of Cyber-Physical Systems
- enable design of CPS through composition of heterogeneous discrete and continuous models via FMI
- models can be co-simulated to empirically evaluate the CPS’s behaviour
- emphasis on semantic integration – how are we to understand the composition of models described in different languages?
  - DE models: VDM-RT (formal language for real-time controllers)
  - CT models: Modelica, 20-sim
  - architecture: SysML
  - integration: FMI
- in this presentation we focus on our work towards formal semantics for Modelica
Formal semantics for Modelica

- give a semantics for a useful subset of the Modelica language
- via a formal account of Hybrid DAEs
- provide a standard against which tools can be checked
- sheds light on “corner cases” where model behaviour may differ across implementations
- link Modelica to other languages via common theoretical factors
- allow principled model integration via FMI
- allow development of an integrated tool-chain
- semantic model mechanised in Isabelle/HOL proof assistant
- allows proof of soundness properties
  - e.g. soundness of code generator wrt. operational semantics
- we report on preliminary work in this direction
Approach: Unifying Theories of Programming

- a framework for the creation, study, and integration of heterogeneous semantic models
- based on an alphabetised relational calculus – cf. Tarski’s relation algebra
- relations as a versatile model of discrete computation
- programs and models denoted as binary relations between inputs and outputs
- all operators are given a relational semantics
- not only about programming languages – abstract models
  - UTP theories used to isolate individual language aspects
    - the building blocks of a complete language semantics
Overview

Modelica

Hybrid DAE

Unifying Theories of Programming

Isabelle/HOL Theorem Proving

UTP Theories

Object-Orientation

Real Time

Concurrency

Rich-state

Hoare Calculi

Contracts
Isabelle/HOL

- a proof assistant for Higher-Order Logic
- HOL – a functional specification language with logical operators
- Isabelle provides powerful support for proof automation
  - equational rewriting
  - logical deduction
  - automated first-order reasoning (sledgehammer)
- applied to several significant verification projects
  - verification of the L4 microkernel (seL4¹)
- includes a growing library of mechanised mathematics
  - including a library for calculus and (non-standard) analysis
- we are using to mechanise our Modelica UTP semantic models

¹https://sel4.systems/
UTP applications

- UTP has been used previously to give formal semantics to a number of languages
- **Safety-Critical Java**\(^2\) – a fragment of real-time Java with a restricted memory model applicable to safety critical programs
- **Handel-C** – a restricted subset of C used to program FPGAs
- **SysML** – internal block diagrams and state machines
  - clarified points where OMG specification is silent
- previous project **COMPASS**\(^3\) demonstrated use of UTP to underlie a model development tool-chain
  - DE simulator, refinement calculator, theorem prover, and model checker
  - multi-pronged approach to verification

\(^2\)https://www.cs.york.ac.uk/circus/hijac/
\(^3\)http://www.compass-research.eu
Approach to UTP Modelica semantics

1. fix a UTP theoretical model of Hybrid Relations
2. define a kernel language of hybrid modelling constructs
   ▶ denotational semantics in UTP model
   ▶ future operational semantics
3. define a mapping from Modelica into kernel language (TODO)
   ▶ give account to event handling mechanism
Semantic Model: Hybrid Relations

- first step is to fix a semantic domain for Hybrid DAEs
- a program in UTP is modelled as a predicate relating
  - input variables \(x\)
  - with output variables \(x'\)
- we extend this with support for continuous variables
  - \(x : \mathbb{R} \rightarrow \mathbb{R}\)
- continuous variables are trajectory functions described by ODEs and DAEs
- each continuous variable \(x\) is accompanied by discrete variables \(x, x'\) denoting its value at the beginning and end of the current trajectory
- **Hybrid Relation** – extension of discrete computation with continuous evolution
Hybrid programs

- formalise key concepts of hybrid behaviour towards Modelica
- we begin with a preliminary kernel language of imperative hybrid programs
- related to formalisms like Hybrid Automata, KeyMaera etc.
- inspired by Hybrid CSP and Duration Calculus
- discrete programming operators
  - sequential composition – $P ; Q$
  - assignment – $x := v$
  - if-then-else conditional – $P \triangleright b \triangleright Q$
  - iteration – $P^*$ and $P^\omega$
- continuous modelling operators
  - interval (continuous invariant) – $[P]$
  - ODEs (TODO DAEs) – $\langle \dot{v}_1 = f_1; \ldots; \dot{v}_n = f_n \rangle$
  - pre-emption – $P [B] Q$
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 $t = [0, 5]$ s
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

\[ p, v := 2, 0 ; (\langle \dot{p} = v; \dot{v} = -9.81 \rangle \left[ p \leq 0 \right] v := -v * .8 ) \]
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;
    elseif x > 21 then
      on = false;
    end when;

    der(x) = if on then 5 - 0.1*x else -0.1*x;

end Thermostat;
```
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;
    elseif x > 21 then
        on = false;
    end when;
    der(x) = if on then 5 - 0.1*x else -0.1*x;
end Thermostat;
```

Thermostat in hybrid relational calculus

\[
x := 20 \; ; \; on := false \; ;
((\langle \dot{x} = 5 - 0.1x \rangle \land on \land \langle \dot{x} = -0.1x \rangle) \\
[x < 19 \lor x > 21](on := true \land x < 19 \lor on := false))^{\omega}
\]
Hybrid Relations in Isabelle/HOL

lift_definition hAssign :: "(('a × 'c ⇒ 'a × 'c) ⇒ ('a, 'c) hrel) is "
  "λ f. {s, s'}. disc s' = f (disc s) ∧ time s' > time s ∧ cont s' = cont s ∧ isHCT2(s, s')}"
by auto

lift_definition hODE ::
  "(('a × 'c ⇒ real × 'c) ⇒ ('a, 'c) hrel) " Hulu"
  is "λ f'. {s, s'}. (∀ τ ∈ {time s ..< time s').
    ((cont s) has_vector_derivative f' (τ, cont s)).
    (at τ within {time s ..< time s'}).)
    ∧ cont s = cont s' ∧ isHCT1(s, s') ∧ isHCT2(s, s')"
by auto

lift_definition hInit :: "('c ⇒ bool) ⇒ ('a, 'c) hrel is
  "λ P. {s, s'}. P (cont s (time s)) ∧ cont s = cont s' ∧ isHCT1(s, s') ∧ isHCT2(s, s')" by auto

lift_definition hFinal :: "('c ⇒ bool) ⇒ ('a, 'c) hrel is
  "λ P. {s, s'}. P (cont s (time s)) ∧ cont s = cont s' ∧ isHCT1(s, s') ∧ isHCT2(s, s')" by auto

definition hPreempt ::
  "('a, 'c) hrel ⇒ ('c ⇒ bool) ⇒ ('a, 'c) hrel ⇒ ('a, 'c) hrel"
(infixr "\_H" 90) where
  "P [B]h Q = (Q \_ hInit B \_ ((λ (τ, v). ¬ B v)) \_ hFinal \_ P) ∧ (\_ hFinal \_ B \_ P) ∧ (\_ hInit \_ Q)"
Next steps

- define translation from Modelica into kernel language
- extend kernel language as necessary
- implement theorem proving facilities for kernel language
- derive operational semantics
- link Modelica semantics to separately defined FMI semantics (in INTO-CPS)
- give semantics to other INTO-CPS languages (VDM-RT, SysML etc.)
Conclusion

- formal semantics are important to gain precise insight into a language
- we started with the definition of kernel language for hybrid modelling constructs based on UTP theories that will continue to evolve as necessary
- within INTO-CPS the goal is to provide sound semantic integration of different languages within a co-simulation framework
- For reaching that goal we aim to provide formal semantics for a suitable subset of Modelica over the next two years
INTO-CPS

INTO-CPS H2020 Grant Agreement 644047

Website: http://into-cps.au.dk/
Follow @IntoCps

Thank you for listening!