INTO-CPS

Towards Verification of Cyber-Physical Systems with UTP and Isabelle/HOL

Simon Foster Jim Woodcock

University of York

University of Oxford

9th January 2017

INTO-CPS

into-cps.au.dk

Horizon 2020 Programme

Outline

Background

Mechanising the UTP

Theory of Hybrid Systems

Conclusion

Outline

Cyber-Physical Systems

- networked, real-time, and embedded systems able to interact with their environment using sensors and actuators
- e.g. robots, self-driving cars
- trustworthiness an important precursor to their adoption
- need for techniques for verifying CPS via formal models
- INTO-CPS project building tools for MBD of CPSs
- emphasis on semantic integration of models
- how to compose heterogeneous "multi-models"
  - SysML for high-level system modelling
  - VDM-RT and Circus for modelling discrete controllers
  - Modelica, 20-sim, and Simulink for system dynamics

Agricultural Robot (AgroIntelli)

- kinematics model (torque, power transfer)
- sensors, camera model
- controller (decision making, network link)
- environment model (field, crops, obstacles)

Horizon 2020 Programme

Unifying Theories of Programming

INTO-CPS

- how to integrate heterogeneous modelling languages and tools?
- study the underlying computational theories in isolation
- UTP theories as building blocks of heterogeneous languages
- supermarket: select & compose theories to produce semantics
- our approach to the multi-model problem

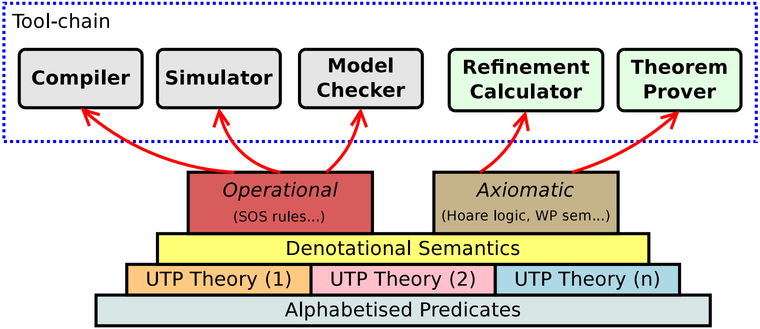
Unifying Theories of Programming

INTO-CPS

- how to integrate heterogeneous modelling languages and tools?
- study the underlying computational theories in isolation
- UTP theories as building blocks of heterogeneous languages
- supermarket: select & compose theories to produce semantics
- our approach to the multi-model problem

"At present, the main available mechanised mathematical tools are programmed for use in isolation [...] it will be necessary to build within each tool a structured library of programming design aids which take the advantage of the particular strengths of that tool. To ensure the tools may safely be used in combination, it is essential that these theories be unified." — ***Chapter 0, UTP book***

UTP semantic stack



UTP background

- environment-free approach to encoding denotational semantics
- based on programs-as-predicates
- programming constructs denoted by characteristic predicates
- predicates encode the set of observable behaviours
- alphabetised relations over input (x), output variables (x')
- alphabet gives the domain of possible observations
- UTP theories encapsulate domains with a set of invariants
- e.g. $time, time' : \mathbb{R}_{\geq 0}$ with $time \leq time'$

INTO-CPS

Example

- simple imperative programming language

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

Algebraic laws of programs

INTO-CPS

$$
\begin{aligned}
(P \,;\, Q) \,;\, R &= P \,;\, (Q \,;\, R) \\
P \,;\, \mathbf{false} &= \mathbf{false} \,;\, P = \mathbf{false} \\
(P \lhd b \rhd Q) \,;\, R &= (P \,;\, R) \lhd b \rhd (Q \,;\, R) \\
\mathbf{while}\ b\ \mathbf{do}\ P &= (P \,;\, \mathbf{while}\ b\ \mathbf{do}\ P) \lhd b \rhd I\!I \\
(P \land b) \,;\, Q &= P \,;\, (b' \land Q) \\
(x := e \,;\, y := f) &= (y := f \,;\, x := e)^{(1)} \\
x := e \,;\, P &= P[e/x]
\end{aligned}
$$

(1) $x \neq y, x \notin fv(f), y \notin fv(e)$

Horizon 2020 Programme

Our work

1. mechanised theory engineering for the UTP framework
   - formalising UTP theories and associated laws
   - transcribe and verify whiteboard-style proofs
   - heterogeneous program verification / refinement

Our work

1. mechanised theory engineering for the UTP framework
   - formalising UTP theories and associated laws
   - transcribe and verify whiteboard-style proofs
   - heterogeneous program verification / refinement
2. creation of new theories to underlie Cyber-Physical Systems
   - reactive processes (CSP)
   - timed reactive designs
   - hybrid relations
   - integration with math libraries (e.g. ODEs)

Outline

Background

Mechanising the UTP

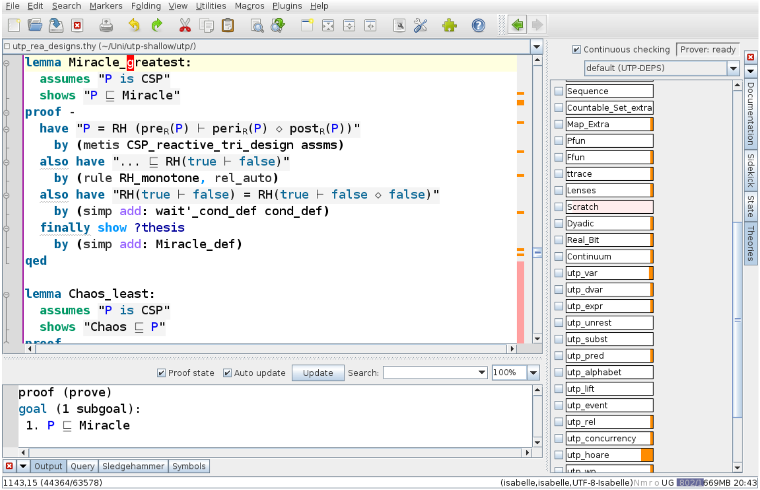
Theory of Hybrid Systems

Conclusion

Isabelle/UTP

- a semantic embedding of the UTP in Isabelle/HOL
  - Isabelle = highly extensible + trustworthy proof framework
- no explicit reliance on syntax – purely denotational approach
- tactics for automating proof steps via HOL proof methods
- large library of algebraic laws of programming
- more than 2000 supporting theorems, lemmas, and proofs
- integration with existing libraries (e.g. lattices, Kleene algebra)

Screenshot


File Edit Search Markers Folding View Utilities Macros Plugins Help
utp_rea_designs.thy (~/Uni/utp-shallow/utp/)
lemma Miracle_greatest:
assumes "P is CSP"
shows "P ⊑ Miracle"
proof -
have "P = RH (preR(P) ⊢ periR(P) ⋄ postR(P))"
by (metis CSP_reactive_tri_design assms)
also have "... ⊑ RH(true ⊢ false)"
by (rule RH_monotone, rel_auto)
also have "RH(true ⊢ false) = RH(true ⊢ false ⋄ false)"
by (simp add: wait'_cond_def cond_def)
finally show ?thesis
by (simp add: Miracle_def)
qed
lemma Chaos_least:
assumes "P is CSP"
shows "Chaos ⊑ P"
Proof state
Auto update
Update
Search:
100%
proof (prove)
goal (1 subgoal):
1. P ⊑ Miracle
Output
Query
Sledgehammer
Symbols
1143,15 (44364/63578)
(isabelle,isabelle,UTF-8-Isabelle)
669MB 20:43
Continuous checking
Prover: ready
default (UTP-DEPS)
Sequence
Countable_Set_extra
Map_Extra
Pfun
Ffun
ttrace
Lenses
Scratch
Dyadic
Real_Bit
Continuum
utp_var
utp_dvar
utp_expr
utp_unrest
utp_subst
utp_pred
utp_alphabet
utp_lift
utp_event
utp_rel
utp_concurrency
utp_hoare
Documentation
Sidekick
State
Theories

Mechanising state spaces

INTO-CPS

- fundamental to the programs-as-predicates approach
- predicates are sets of observations of the state
- should combine efficient proof automation with expressivity
- deep vs. shallow embedding dichotomy

Mechanising state spaces

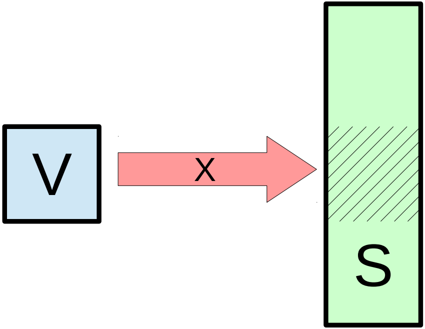
INTO-CPS

- fundamental to the programs-as-predicates approach
- predicates are sets of observations of the state
- should combine efficient proof automation with expressivity
- deep vs. shallow embedding dichotomy

- lenses as a uniform semantic interface for variables
- identify variables by the position they occupy in the state
- regions of the state can be variously composed and related
- using separation algebra style operators
- nameless and spatial representation of variables

Lenses

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S



Lenses

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised by intuitive laws

$$\mathit{get}\,(\mathit{put}\,s\;v) = v \qquad \text{(PutGet)}$$

$$\mathit{put}\,(\mathit{put}\,s\;v')\;v = \mathit{put}\,s\;v \qquad \text{(PutPut)}$$

$$\mathit{put}\,s\;(\mathit{get}\,s) = s \qquad \text{(GetPut)}$$

Lenses

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \rightarrow V$
  - $\mathit{put} : S \rightarrow V \rightarrow S$
- characterised by intuitive laws
- several models, example: record lenses

$$\llparenthesis\ \mathit{forename} : \mathit{String}, \mathit{surname} : \mathit{String}, \mathit{age} : \mathit{Int}\ \rrparenthesis$$

INTO-CPS

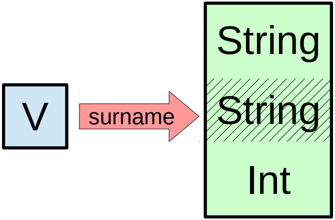
Lenses

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised by intuitive laws
- several models, example: record lenses

$$\lparr \underbrace{\mathit{forename} : String}_{\text{lens 1}}, \underbrace{\mathit{surname} : String}_{\text{lens 2}}, \underbrace{\mathit{age} : Int}_{\text{lens 3}} \rparr$$

Lenses

- $X : V \Longrightarrow S$ for view type V and ("bigger") source type S
- allow to focus on V independently of rest of S
- signature consists of two functions:
  - $\mathit{get} : S \to V$
  - $\mathit{put} : S \to V \to S$
- characterised by intuitive laws
- several models, example: record lenses



Lens comparison

- how to compare the behaviour of two lenses?

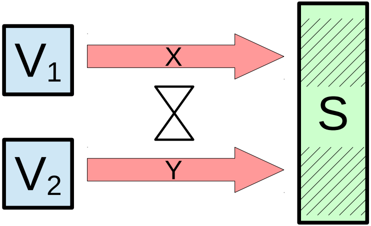
Lens comparison

- how to compare the behaviour of two lenses?
- e.g. are two variables (behaviourally) identical?

Lens comparison

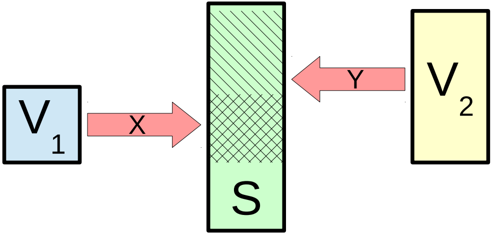
- how to compare the behaviour of two lenses?
- e.g. are two variables (behaviourally) identical?

- lens independence ($X \bowtie Y$)
- X, Y are independent if they view spatially separate regions
- e.g. $forename \bowtie surname$



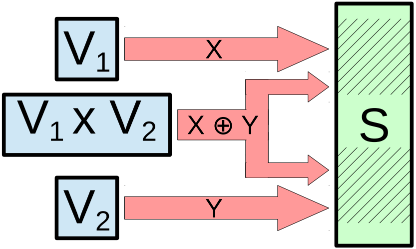
Sublens relation

- if lenses X and Y are not independent, how are they related?
- X is a sublens of Y ($X \preceq Y$) if Y's view encompasses X's
- can also induce an equivalence $X \approx Y \Leftrightarrow X \preceq Y \wedge Y \preceq X$



Lens sum

- $X \oplus Y$ parallel composes two independent lenses
- similar to the heap composition operator of separation algebra



- e.g. $X \preceq X \oplus Y$ and $X \oplus Y \approx Y \oplus X$

Mechanised alphabetised predicates

- alphabets are modelled as Isabelle types (α)
- our basic predicate model is $\mathbb{P}\,\alpha$
- lenses $\tau \Longrightarrow \alpha$ model the variables
- variable sets using $\oplus$ for $\cup$
- predicate operators created by lifting Isabelle equivalents
- provides direct proof automation support from HOL libraries
- can denote meta-logical style operators:
  - $x \sharp P$ – P does not depend on lens x
  - $P[v/x]$ – assign v to lens x
  - $P \oplus_p a$ – extend alphabet using lens $a : \alpha \Longrightarrow \beta$
- from this basis we prove the UTP laws of programming

Laws of programming

Theorem (Unital quantale)

UTP relations form a unital quantale and thus a Kleene algebra (Armstrong, 2015)

Laws of programming

INTO-CPS

Theorem (Unital quantale)

UTP relations form a unital quantale and thus a Kleene algebra (Armstrong, 2015)

Theorem (Assignment laws)

$$
\begin{array}{rcll}
x := e \,;\, P & = & P[e/x] & \\
x := e \,;\, x := f & = & x := f & x \sharp f \\
x := e \,;\, y := f & = & y := f \,;\, x := e & x \bowtie y, x \sharp f, y \sharp e
\end{array}
$$

Outline

Background

Mechanising the UTP

Theory of Hybrid Systems

Conclusion

Hybrid Systems in UTP

INTO-CPS

- support semantics for languages like Modelica and Simulink
- augment UTP's relational calculus with continuous variables
  - modelled as partial contiguous functions $\underline{x} : \mathbb{R}_{\geq 0} \nrightarrow \mathbb{R}$

Hybrid Systems in UTP

INTO-CPS

- support semantics for languages like Modelica and Simulink
- augment UTP's relational calculus with continuous variables
  - modelled as partial contiguous functions $\underline{x} : \mathbb{R}_{\geq 0} \nrightarrow \mathbb{R}$
- retain standard discrete operators defs – e.g. $P \,;\, Q$, $x := v$
- combine discrete and continuous with coupling invariants
- combine with other theories, e.g. CSP and reactive designs

Hybrid Systems in UTP

- support semantics for languages like Modelica and Simulink
- augment UTP's relational calculus with continuous variables
  - modelled as partial contiguous functions $\underline{x} : \mathbb{R}_{\geq 0} \nrightarrow \mathbb{R}$
- retain standard discrete operators defs – e.g. $P \,;\, Q$, $x := v$
- combine discrete and continuous with coupling invariants
- 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

- kernel language of imperative hybrid programs
- operators given a semantics in the theory of hybrid relations

Hybrid relational calculus

INTO-CPS

- kernel language of imperative hybrid programs
- operators given a semantics in the theory of hybrid relations
- discrete relational operators
  - sequential composition — $P \; ; \; Q$
  - assignment — $x := v$
  - if-then-else conditional — $P \lhd b \rhd Q$
  - iteration — P^* and P^ω

Hybrid relational calculus

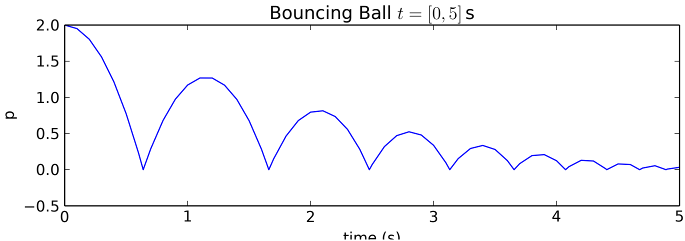
INTO-CPS

- kernel language of imperative hybrid programs
- operators given a semantics in the theory of hybrid relations
- discrete relational operators
  - sequential composition — $P \; ; \; Q$
  - assignment — $x := v$
  - if-then-else conditional — $P \lhd b \rhd Q$
  - iteration — P^* and P^ω
- continuous evolution operators
  - differential equations — $\langle \underline{\dot{x}} = \mathcal{F}(\underline{x}, \underline{\dot{x}}) \rangle$
  - pre-emption — $P \; [\; B \;] \; Q$
  - interval (continuous invariant) — $\llbracket P \rrbracket$

Example: 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: 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

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

UTP theory of hybrid relations

- model for hybrid relational calculus
- use timed trace model from Hayes et al.

UTP theory of hybrid relations

INTO-CPS

- model for hybrid relational calculus
- use timed trace model from Hayes et al.
- hybrid relation = set of possible continuous variable evolutions
  - e.g. $x := v \; ; \langle \dot{x} = \mathcal{F}(\underline{x}, \underline{\dot{x}}) \rangle$ sets up initial value problem

UTP theory of hybrid relations

- model for hybrid relational calculus
- use timed trace model from Hayes et al.
- hybrid relation = set of possible continuous variable evolutions
  - e.g. $x := v \,;\, \langle\, \dot{x} = \mathcal{F}(\underline{x}, \underline{\dot{x}})\, \rangle$ sets up initial value problem
- $\ell : \mathbb{R}_{\geq 0}$ length of the current observation
- open domain for continuous variables $\mathrm{dom}(\underline{x}) = [0, \ell)$

UTP theory of hybrid relations

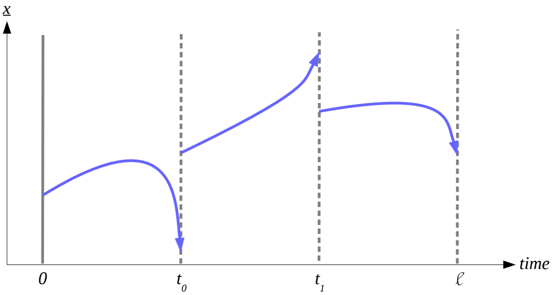
- model for hybrid relational calculus
- use timed trace model from Hayes et al.
- hybrid relation = set of possible continuous variable evolutions
  - e.g. $x := v \,;\, \langle\, \dot{x} = \mathcal{F}(\underline{x}, \underline{\dot{x}}) \,\rangle$ sets up initial value problem
- $\ell : \mathbb{R}_{\geq 0}$ length of the current observation
- open domain for continuous variables $\mathrm{dom}(\underline{x}) = [0, \ell)$
- no continuous variable sharing in sequence $P \,;\, Q$
- continuous variables accompanied by relational "copy variables"
- continuous linking invariants: $x = \underline{x}(0)$ and $x' = \lim_{t \to \ell}(\underline{x}(t))$

UTP theory of hybrid relations

INTO-CPS

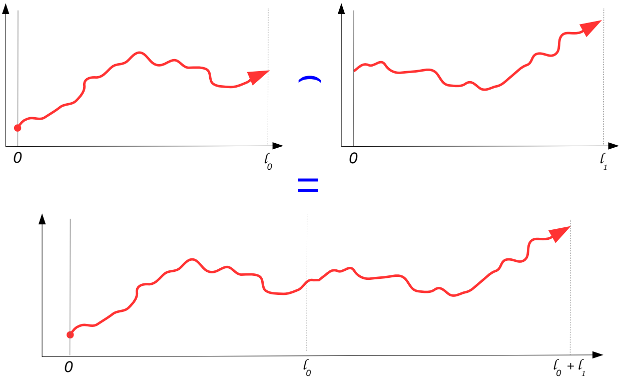
- model for hybrid relational calculus
- use timed trace model from Hayes et al.
- hybrid relation = set of possible continuous variable evolutions
  - e.g. $x := v \; ; \langle \dot{x} = \mathcal{F}(\underline{x}, \underline{\dot{x}}) \rangle$ sets up initial value problem
- $\ell : \mathbb{R}_{\geq 0}$ length of the current observation
- open domain for continuous variables $\mathrm{dom}(\underline{x}) = [0, \ell)$
- no continuous variable sharing in sequence $P \; ; \; Q$
- continuous variables accompanied by relational "copy variables"
- continuous linking invariants: $x = \underline{x}(0)$ and $x' = \lim_{t \to \ell}(\underline{x}(t))$
- observational variables: $tr, tr' : \mathbb{TT}$ – the timed trace

Timed traces ($\mathbb{TT}$)



- represent relative continuous evolution of a process
- finite set of discontinuities represent events (e.g. assignment)
- domain is right-open interval $[0, \ell)$
- range is a suitable topological space Σ – the continuous state
- allows integration of non-continuous data (e.g. CSP events)
- continuous variables as lenses on Σ

Trace concatenation



- theorem: closed under piecewise continuity and convergence
- partial order: $x \leq y \Leftrightarrow \exists z \bullet y = x \frown z$
- subtraction: $x - y \triangleq \iota z \bullet y = x \frown z$

Hybrid denotational semantics

INTO-CPS

$$\lceil P \rceil \triangleq tr' > tr \wedge (\forall\, t \in [0, \ell) \bullet P[\underline{x}(t)/\underline{x}])$$

$$[\![P]\!] \triangleq \lceil P \rceil \wedge \left(x = \underline{x}(0) \wedge v' = \lim_{t \to \ell} (\underline{x}(t)) \right)$$

$$\langle\, \dot{\underline{x}} = \mathcal{F}'(\underline{x}, \dot{\underline{x}}) \,|\, \mathcal{B}(\underline{x}) \,\rangle \triangleq \exists\, \mathcal{F} \bullet [\![\mathcal{F}' \text{ has-deriv } \mathcal{F} \text{ at } \tau \wedge \underline{x} = \mathcal{F}(\tau) \wedge \mathcal{B}(\underline{x})]\!]$$

Hybrid denotational semantics

INTO-CPS

$$\lceil P \rceil \triangleq tr' > tr \wedge (\forall\, t \in [0, \ell) \bullet P[\underline{x}(t)/\underline{x}])$$

$$\llbracket P \rrbracket \triangleq \lceil P \rceil \wedge \left(x = \underline{x}(0) \wedge v' = \lim_{t \to \ell} (\underline{x}(t)) \right)$$

$$\langle\, \underline{\dot{x}} = \mathcal{F}'(\underline{x}, \underline{\dot{x}}) \,|\, \mathcal{B}(\underline{x}) \,\rangle \triangleq \exists\, \mathcal{F} \bullet \llbracket \mathcal{F}' \text{ has-deriv } \mathcal{F} \text{ at } \tau \wedge \underline{x} = \mathcal{F}(\tau) \wedge \mathcal{B}(\underline{x}) \rrbracket$$

- current work focuses on defining CSP operators
- takes inspiration from Hybrid CSP and Duration Calculus

Mechanisation

- based on Isabelle/UTP and the Multivariate Analysis package
- proof support for hybrid relational calculi
- real numbers based on Cauchy sequences
- differential equations based on topological and metric spaces
- support for limits, ODEs, and their solutions
- proved key properties of healthiness conditions and signature
- continuous variables as lenses into Σ

Outline

Horizon 2020 Programme

Conclusion

- UTP is a great platform for integrating diverse semantics
- we are applying it to production of formal semantics that underlie Cyber-Physical Systems
- all of which are being mechanised in Isabelle/UTP
- https://github.com/isabelle-utp/utp-main
- enables integration of reactive and hybrid systems
- applying these techniques to INTO-CPS case studies
- long-term aim proof support for CPS
- through integration with other tools (e.g. CAS)
- and a new extension of Circus called CyPhyCircus

References

INTO-CPS

- A. Cavalcanti and J. Woodcock. A Tutorial Introduction to CSP in Unifying Theories of Programming. PSSE 2004. LNCS 3167. pp. 220–268.
- S. Foster, B. Thiele, A. Cavalcanti, and J. Woodcock. Towards a UTP semantics for Modelica. UTP 2016. LNCS.
- S. Foster, F. Zeyda, and J. Woodcock. Unifying heterogeneous state-spaces with lenses. ICTAC 2016. LNCS 9965.
- C. Zhou, A. P. Ravn, M. R. Hansen. An extended Duration Calculus for hybrid real-time systems. Hybrid Systems. LNCS 736. pp. 36–59. 1993.
- I. J. Hayes, S. E. Dunne, and L. Meinicke. Unifying theories of programming that distinguish nontermination and abort. MPC 2010. LNCS 6120. pp. 178–194.
- C. Zhou, J. Wang, A. P. Ravn. A formal description of hybrid systems. Hybrid Systems III: Verification and Control. LNCS 1066. pp. 511-530. 1995.
- J. He. HRML: a hybrid relational modelling language. QRS 2015.