A UTP theory for Hybrid Computation

Simon Foster

Friday 19th October, 2018
Outline

1. Background
2. Foundations
3. Hybrid Relations
4. Proof Support
5. Conclusion
Outline

1. Background
2. Foundations
3. Hybrid Relations
4. Proof Support
5. Conclusion
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

\[
x := v \triangleq x' = v \land y' = y
\]

\[
P ; Q \triangleq \exists x_0 \bullet P[x_0/x'] \land Q[x_0/x]
\]

\[
P \triangleright b \triangleright Q \triangleq (b \land P) \lor (\neg b \land Q)
\]

\[
P^* \triangleq \nu X \bullet P ; X
\]

- how to go beyond simple imperative behaviour?
- UTP theories to isolate paradigmatic aspects of a language
- compose theories to produce heterogeneous semantic models
UTP Theory of Reactive Processes

- a generic foundation for trace-based reactive programming semantics
- used in the denotational models of CSP, ACP, the Circus family etc.
- key observational variables:
  - \( \text{wait, wait}' : \mathbb{B} \) – determines whether process is quiescent
  - \( tr, tr' : \text{seq Event} \) – the event trace before and after execution

**Definition (Reactive Healthiness Conditions)**

\[
R1(P) \triangleq P \land tr \leq tr'
\]

\[
R2(P) \triangleq P[\langle \rangle, tr' - tr / tr, tr']
\]

\[
R3(P) \triangleq \exists t \cdot P[\langle \rangle, t / tr, tr'] \land tr' = tr \bowtie t
\]

**Theorem (R1-R2 trace contribution)**

\[
R1(R2(P)) = (\exists t \cdot P[\langle \rangle, t / tr, tr'] \land tr' = tr \bowtie t)
\]
Hybrid Systems

- hybrid computations as piecewise-continuous functions
- consider them as a form of trace (cf. Hayes 2006 – timed traces)
- questions – can we
  - unify hybrid and reactive programming theories?
  - construct a semantic model for hybrid programming?
  - build a verification tool for this in Isabelle/HOL?
Outline

1. Background

2. Foundations

3. Hybrid Relations

4. Proof Support

5. Conclusion
Trace Algebra

Definition (Trace Algebra)

A trace algebra \((\mathcal{T}, \leftarrow, \epsilon)\) satisfies the following axioms:

1. \[ x \leftarrow (y \leftarrow z) = (x \leftarrow y) \leftarrow z \quad \text{(TA1)} \]
2. \[ \epsilon \leftarrow x = x \leftarrow \epsilon = x \quad \text{(TA2)} \]
3. \[ x \leftarrow y = x \leftarrow z \implies y = z \quad \text{(TA3)} \]
4. \[ x \leftarrow z = y \leftarrow z \implies x = y \quad \text{(TA4)} \]
5. \[ x \leftarrow y = \epsilon \implies x = \epsilon \quad \text{(TA5)} \]

- also known as positive cancellative monoids
- allows us to characterise generalised reactive processes
Definition (Trace Prefix and Subtraction)

\[ x \leq y \iff \exists z \bullet y = x \bowtie z \]

\[ y - x \triangleq \begin{cases} 
\iota z \bullet y = x \bowtie z & \text{if } x \leq y \\
\epsilon & \text{otherwise}
\end{cases} \]

- example prefix: \( \langle a, b, c \rangle \leq \langle a, b, c, d, e \rangle \)
- example subtraction: \( \langle a, b, c, d, e \rangle - \langle a, b, c \rangle = \langle d, e \rangle \)

Theorem (Trace Prefix Laws)

\((T, \leq) \) is a partial order

(TP1) 
\( \epsilon \leq x \)

(TP2) 
\( x \leq x \bowtie y \)

(TP3) 
\( x \bowtie y \leq x \bowtie z \iff y \leq z \)
Theorem (Trace Subtraction Laws)

\[
\begin{align*}
x - \epsilon &= x \\
\epsilon - x &= \epsilon \\
x - x &= \epsilon \\
(x \Join y) - x &= y \\
(x - y) - z &= x - (y \Join z) \\
(x \Join y) - (x \Join z) &= y - z \\
y \leq x &\land x - y = \epsilon \iff x = y \\
x \leq y &\implies x \Join (y - x) = y
\end{align*}
\]
Trace Algebra Models

Theorem
\((\mathbb{R}_{\geq 0}, +, 0)\) forms a trace algebra.

Theorem
\((\text{seq Event}, \sqcap, \langle \rangle)\) forms a trace algebra

Theorem
\((\mathbb{T}_\Sigma, \sqcap, \epsilon)\) forms a trace algebra
Timed Traces

Definition (Timed Traces)

\[ \mathbb{T}_\Sigma \triangleq \left\{ f : \mathbb{R}_{\geq 0} \to \Sigma \mid \exists t : \mathbb{R}_{\geq 0} \bullet \text{dom}(f) = [0, t) \land t > 0 \Rightarrow \exists I : \mathbb{R}_{\text{oseq}} \right. \]
\[ \left. \quad \text{ran}(I) \subseteq [0, t] \land \{0, t\} \subseteq \text{ran}(I) \land \left( \forall n < \#I - 1 \bullet f \text{ cont-on } [I_n, I_{n+1}] \right) \right\} \]

\[ \mathbb{R}_{\text{oseq}} \triangleq \{ x : \text{seq } \mathbb{R} \mid \forall n < \#x - 1 \bullet x_n < x_{n+1} \} \]

\[ f \text{ cont-on } [m, n) \triangleq \forall t \in [m, n) \bullet \lim_{x \to t^+} f(x) = f(t) \]

- a timed trace \( t \) is a sequence of continuous segments over \([0, \text{end}(t))\)
- state space \( \Sigma \) must form a Hausdorff topological space (minimally)
- concatenation \( x \bowtie y \) shifts \( y \) to the right and takes the union
- model and laws all mechanised in Isabelle/UTP
Trace Append

\[ 0 \oplus l_0 = \sim \]

\[ \sim \]

\[ \sim \]

\[ l_0 + l_1 \]
Generalised Reactive Processes

Definition (Reactive Healthiness Conditions)

\[ R_1(P) \triangleq P \land tr \leq tr' \]
\[ R_2(P) \triangleq P[\langle \rangle, tr' - tr / tr, tr'] \]
\[ R_3(P) \triangleq \text{II} \circ \text{wait} \circ P \]
\[ R(P) \triangleq R_1 \circ R_2 \circ R_3 \]

- In this generalised setting, all key theorems are retained:
  - \( R \) is idempotent and monotonic (and continuous)
  - \( R \) forms a complete lattice under \( \sqsubseteq \)
  - \( R \) closed under relational operators (\( \text{II}, ;, x := v \))

- This provides the foundation for denotational semantics of hybrid systems
Outline

1. Background

2. Foundations

3. Hybrid Relations

4. Proof Support

5. Conclusion
Hybrid Relational Calculus

- extend relational calculus with **continuous variables** ($\tilde{x}(t)$)
- by embedding them into the **timed trace model**
- each $\tilde{x} : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ is a piecewise continuous function
- hybrid relation = set of possible ways variables can evolve
- support for **discrete assignment** to continuous variables

operators:

- relational operators (sequence, assignment, if-then-else, etc.)
- function evolution — $\tilde{x}(t) \leftarrow f(x_0, t)$
- differential equations — $\langle (\dot{x}_1, \cdots, \dot{x}_n) \bullet (f_1, \cdots, f_n) \rangle$
- interruption — $P \triangle \langle b \mid c \rangle$
Example: Bouncing Ball

Continuous variables:
- \( h : \mathbb{R} \) – ball’s height above the ground, and \( v : \mathbb{R} \) – velocity

System definition:

\[
\text{BBall} \triangleq h := 2.0 ; v := 0 ;
\]

\[
\begin{align*}
\left( \langle \dot{h}, \dot{v} \rangle \bullet (v, -9.81) \right) \\
\triangle \langle h \geq 0 \mid v \leq 0 \land h < \epsilon \rangle ; \\
v := -0.8 \cdot v
\end{align*}
\]

Simulated Behaviour:
Denotational Semantics

- \( tr, tr': T T_\Sigma \) – timed trace before and after execution
- \( st, st': \Sigma \) – discrete state before and after execution
- Continuous variables are lenses, e.g. \( x: \mathbb{R} \rightarrow \Sigma \)

**Definition (Continuous Variable Operators)**

\[
\begin{align*}
\mathbf{tt} & \triangleq tr' - tr \\
\ell & \triangleq \text{end}(\mathbf{tt}) \\
\tilde{x}(t) & \triangleq \mathbf{tt}(t):x \\
P @ t & \triangleq P[\mathbf{tt}(t)/v'] \\
\lceil P(t) \rceil & \triangleq R1 (\forall t \in [0, \ell) \bullet P(t) @ t)
\end{align*}
\]

- \( P @ t \) – the predicate \( P \) is true at instant \( t \)
- \( P \) is relation – \( x \) is initial value \((x_0)\); \( x' \) at each instant \( t \)
  - e.g. \((x' > (y' + 10 - x)) @ 7 \implies \tilde{x}(7) > (\tilde{y}(7) + 10 - x)\)
- \( \lceil P(t) \rceil \) is a generalised duration calculus like interval
Continuous Functions

**Definition (Function Evolution)**

\[ \tilde{x}(t) \leftarrow f(t, x) \triangleq ([x' = f(t, x)] \land \ell > 0) \]

\[ \tilde{x}(t) \leftarrow f(t, x) \triangleq (\tilde{x}(t, x) \leftarrow f(t) \land \ell \leq d) \]

where \( x : \mathbb{R}^n \implies \Sigma, f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n, \) and \( d : \mathbb{R}_{\geq 0} \)

**Example**

\[ \tilde{v}(t) \leftarrow v - 9.81 \cdot t \]

\[ = ([v' = v - 9.81 \cdot t] \land \ell > 0) \]

\[ = (R1(\forall t \in [0, \ell) \bullet \tilde{v}(t) = v - 9.81 \cdot t) \land \ell > 0) \]

\[ = (tr \leq tr' \land (\forall t \in [0, \ell) \bullet tt(t) : v = v - 9.81 \cdot t) \land \ell > 0) \]
Semantics of Assignment

Example

\[ x := 1 ; x := x + 1 ; x := x + 2 \]

- what is the value of \( \tilde{x}(\ell) \)? 1, 2, or 4?
- problem: a variable can only have one value at each instant
- we maintain discrete copies of each continuous variables
- assignment is then simply relational assignment (\( x' = 4 \))
- continuous variables only updated during evolution
- we will use assignments to provide initial conditions for ODEs
- coupling invariants link discrete and continuous
Ordinary Differential Equations

Definition (ODE in Hybrid Relations)

\[ \langle \dot{x} \bullet f(t, x) \rangle \triangleq \left( \exists g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n, l : \mathbb{R}_{\geq 0} \\
\bullet \quad \ell > 0 \land \ell = l \land x = \tilde{x}(0) \\
\bullet \quad \land [g \text{ has-ode-deriv } f \text{ at } (t < l) \land x' = g(t)] \right) \]

where \( x : \mathbb{R}^n \rightarrow \Sigma \) and \( f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n \rightarrow \mathbb{R}^n \) (system of ODEs)

- intuition: there exists solution \( g \) for the IVP composed of \( f \) and \( x \)
- only permit evolutions of a non-zero length (\( \ell > 0 \))
- \( g \text{ has-ode-deriv } f \text{ at } (t < l) \) – derivative of \( g \) on \([0, l)\) is \( f \)
- formula \( x = \tilde{x}(0) \) links discrete copy to the continuous at \( t = 0 \)

Example (Fully Specified IVP)

\( (h, v) := (2.0, 0) ; \langle (\dot{h}, \dot{v}) \bullet (v, -9.81) \rangle \)
Solving Differential Equations

**Theorem**
If, for any $v : \mathbb{R}^n$ and $l > 0$, $g(v)$ is the unique solution to $f$ on the interval $[0, l]$, and $g(v, 0) = v$ then $\langle \dot{x} \cdot f(t, x) \rangle = x \leftarrow g(x, t)$.

**Example**

$$\langle (h, v) \cdot (v, -9.81) \rangle$$

$$\downarrow$$

$$(h, v) \leftarrow \left( \frac{-9.81 \cdot t^2}{2} + v \cdot t + h, -9.81 \cdot t + v \right)$$
Interruption

Definition (Preemption)

\[ P \triangle \langle b \mid c \rangle \triangleq (P \land [b] \land \ell > 0 \land (v' = \lim_{t \to \ell^-} \tilde{v}(t)) \land c') \]

- \( P \) is permitted to evolve while ever invariant \( b \) is satisfied
- when \( c \) becomes true the evolution terminates producing a final value
- expression \( v : \Sigma \) refers to the entire continuous state
- the final value of \( v \) takes the left-limit of \( \tilde{v} \)

\[
\begin{align*}
\text{BBall} & \triangleq h := 2.0 ; v := 0 ; \\
& \quad \left( (\langle \dot{h}, \dot{v} \rangle \bullet (v, -9.81) \right) \\
& \quad \triangle \langle h \geq 0 \mid v \leq 0 \land h < \epsilon \rangle ; \\
& \quad v := -0.8 \cdot v
\end{align*}
\]
Solving a preempted function evolution

**Theorem (Evolution Termination)**

We assume that $f$ is a continuous function on the domain $[0, l]$, where $l > k$ and $k > 0$, and the following conditions hold:

1. $b$ is satisfied for all instants $t \in [0, l) : \forall t \in [0, l) \bullet b[f(t)/x']$.
2. $b$ becomes false at $l : \neg b[f(l)/x']$.
3. $c$ is not satisfied for all instants $t \in [0, k) : \forall t \in [0, k) \bullet \neg c[f(t)/x']$.
4. $c$ becomes true at $k$ and stays true until $l : \forall t \in [k, l) \bullet c[f(t)/x']$.

Then the following equality holds:

$$(x \leftarrow f(t_i) \triangle \langle b \mid c \rangle) = \left( x \leftarrow [k\ldots l] f(t_i) \right)$$

- can turn a preempted solvable ODE into a bounded function evolution
Outline

1. Background
2. Foundations
3. Hybrid Relations
4. Proof Support
5. Conclusion
Verification of Hybrid Systems

- **hybrid relations**: extend relational calculus with continuous variables
- **differential equations and analysis supported by** HOL-ODE

```verbatim
definition BrakingTrain =
  (c:accel, c:vel, c:pos) :=, («normal_deceleration», «max_speed», «0») ;;
  {&accel,&vel,&pos} • train_ode(ti)_h until_h (vel` ≤_u 0) ;; c:accel ::=, 0"

theorem braking_train_pos_le:
  "($st:c:accel` =_u 0 ∧ [pos` <_u 44]_h) ⊆ BrakingTrain" (is "?lhs ⊆ ?rhs")
proof -
  — < Solve ODE, replacing it with an explicit solution: @{term train_sol}. >
  have "?rhs =
    (c:accel, c:vel, c:pos) :=, («-1.4», «4.16», «0») ;;
    {&accel,&vel,&pos} ←_h «train_sol»(accel,vel,pos)_a(«ti»)_a until_h (vel` ≤_u 0) ;;
    c:accel ::=, 0"
  by (simp only: BrakingTrain_def train_sol)
  — < Set up initial values for the ODE solution using assigned variables. >
  also have "... =
    {&accel,&vel,&pos} ←_h «train_sol(-1.4,4.16,0)(ti)» until_h (vel` ≤_u 0) ;; c:accel ::=, 0"
  by (rel_auto)
  — < Find the point at which the train stops >
  also have "... =
    ({{&accel,&vel,&pos} ←_h(«416/140») «train_sol(-1.4,4.16,0)(ti)>>}) ;; c:accel ::=, 0"
  apply (literalise)
  apply (subst hUntil_solve[of _ "416/140"])
```
Continuous VCs in Isabelle/HOL

- approximation tactic solves real inequalities (incl. transcendental)

```isabelle
— < Prove that this satisfies the continuous invariant ⟩
also have "?lhs ⊆ ..."
proof (rel_simp)
  fix tr tr' :: "'a cst_train_scheme ttrace" and t::real
  assume "endₜ (tr' - tr) * 35 = 104" "0 ≤ t" "t < endₜ (tr' - tr)"
  hence "t ∈ {0..416/140}"
  by auto
  thus "104 * t / 25 - 7 * t² / 10 < 44"
  by (approximation 4)
qed
```
Outline

1. Background
2. Foundations
3. Hybrid Relations
4. Proof Support
5. Conclusion
Conclusions

- generalised reactive processes to unify with hybrid computation
- provided a model of traces based on piecewise continuous functions
- constructed hybrid relations, with continuous variables and ODEs
- proved several theorems for solving ODEs
- built a prototype verification tool in Isabelle/UTP
- application: denotational semantics for the Modelica language
Future Work

- advanced timed trace models, e.g. superdense time \((\mathbb{R}_{\geq 0} \times \mathbb{N} \rightarrow \mathbb{R}^n)\)
- linking my model to Platzer’s hybrid programs (from KeYmaera)
- integration of differential induction technique (cf. RAMiCS paper by Struth et al.)
- support for concurrency – denotational semantics for HCSP?
- case studies
Resources and Publications

- **Isabelle/UTP**: [https://www.cs.york.ac.uk/circus/isabelle-utp/](https://www.cs.york.ac.uk/circus/isabelle-utp/)
- **CyPhyAssure**: [https://www.cs.york.ac.uk/circus/CyPhyAssure/](https://www.cs.york.ac.uk/circus/CyPhyAssure/)


