Semantic Families for Cyber-physical Systems

Jan Peleska
University of Bremen
Verified Systems International GmbH
jp@cs.uni-bremen.de
2015-12-07

BCS FACS - Annual Peter Landin Semantics Seminar 2015
Overview

• Semantics for CPS – time for a change of paradigm?

• Multiple formalisms in CPS modelling
  - Example 1. Testing theories and collaborative tool environments
  - Example 2. Verification of emergent properties

• Conclusions and future work
Semantics for CPS – time for a change of paradigm?

• Semantics for CPS – time for a change of paradigm?
  • Multiple formalisms in CPS modelling
    • Example 1. Testing theories and collaborative tool environments
    • Example 2. Verification of emergent properties
  • Conclusions and future work

Semantics for CPS – time for a change of paradigm?
Recall

- The investigation of concurrent systems semantics started somewhere in the seventies of the last century.

C. A. R. Hoare: 
Recall

- Since then, a multitude of formalisms has been developed and successfully applied to

- **Development**
  - modelling
  - code generation

- **Verification & Validation**
  - theorem proving
  - model checking
  - simulation
  - testing
Cyber-physical systems

- Systems of collaborating computational elements controlling physical entities

https://en.wikipedia.org/wiki/Cyber-physical_system

Image courtesy of Daimler AG
## Some CPS-characteristics affecting semantic modelling

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution, time-discrete and time-continuous control</td>
<td>Hybrid systems semantics</td>
</tr>
<tr>
<td>Modeling using multiple formalisms</td>
<td>Model, sentence, and theory translation</td>
</tr>
<tr>
<td>Emergent properties</td>
<td>Temporal logic, trace logic – how to verify in presence of multiple formalisms?</td>
</tr>
<tr>
<td>Dynamic re-configuration</td>
<td>Semantics for object-oriented systems – or can we find something simpler?</td>
</tr>
<tr>
<td>Evolution of asserted component behaviours</td>
<td>New paradigms for behavioural assertions</td>
</tr>
<tr>
<td>Large numbers of replicated components</td>
<td>Can the knowledge about replication lead to optimised V&amp;V methods?</td>
</tr>
</tbody>
</table>
Multiple formalisms in CPS modelling – Example 1. Testing theories and collaborative tool environments
Application scenario

• CPS consists of several components

• Some components are modelled by finite state machines (FSMs)

• Other components are modelled by SysML state machines with Kripke structure semantics
Application scenario – train onboard speed control

Onboard main controller

PLC brake controller

Automated braking command

Train engine driver brake command

Current speed

Current maximum speed

Emergency brake

man_on, man_off

auto_on, auto_off

trigger, release
Application scenario – train onboard speed control

Current speed

Current maximum speed

Emergency brake

Train engine driver brake command

man_on, man_off

auto_on, auto_off

trigger, release

RELEASED

auto_off, man_off/release

TRIGGERED

auto_on/trigger auto_off/trigger auto_on/trigger auto_off/release

TRIGGERED_AUTO

man_off, man_on/trigger

stim CSM_OFF

WARNING

+ entry / d = WARNING;

NORMAL

+ entry / d = NORMAL;
+ entry / b = NO_CMD;

Overspeed

+ entry / d = OVERSPEED;

INTERVENTION_LEVEL_1

+ entry / d = INTERVENTION;
+ entry / b = EMER_BRAKE_CMD;

INTERVENTION_LEVEL_2

[\dot{v} > v \_m + \dot{\phi} \_L \_1 \_L \_1 \_v \_m]
Brake controller

- Discrete inputs
- Discrete internal state
- Discrete outputs

Apply **complete FSM testing strategy**
Complete test suites

- Defined with respect to **fault model** \( (M, \leq, Dom) \), that is,
  - a reference model \( M \)
  - a conformance relation \( \leq \)
  - a fault domain \( Dom \)

- **Complete** = sound + exhaustive

- **Sound** = every \( M' \) in \( Dom \) satisfying \( M' \leq M \) passes

- **Exhaustive** = every \( M' \) in \( Dom \) violating \( M' \leq M \) fails
Complete FSM test suites

- For FSMs, many complete testing strategies exist
  - for deterministic or nondeterministic FSMs
  - for completely defined or incomplete FSMs

Alexandre Petrenko, Nina Yevtushenko:
Adaptive Testing of Nondeterministic Systems with FSM. HASE 2014: 224-228

Robert M. Hierons:
Onboard main controller

- Large input domains – speed
- Discrete internal state
- Discrete outputs

⭐Apply input equivalence class testing
⭐Can we also apply a complete strategy?

⭐TTT = Testing Theory Translation
• Consider different **semantic domains** with their conformance relations
  
  ◆ Finite state machines – language equivalence, language containment
  
  ◆ Kripke structures – I/O-equivalence, I/O-refinement

• Fix a **signature** in each domain
  
  ◆ \( Sig_1 \) – Kripke structures over fixed I/O variables
  
  ◆ \( Sig_2 \) – FSMs over fixed I/O-alphabet
• Create a **model map** $T$ from sub-domain of $Sig_1$ to $Sig_2$

$$T : Dom_1 \rightarrow Sig_2;$$
$$Dom_1 \subseteq Sig_1$$

• Create a **test case map** $T^*$ from test cases of $Sig_2$ to test cases of $Sig_1$

$$T^* : TC(Sig_2) \rightarrow TC(Sig_1)$$

• Prove the **satisfaction condition**
Satisfaction condition

**Condition 1.** The model map is compatible with the conformance relations

$$\forall S, S' \in Dom_1 : S' \leq_1 S \Leftrightarrow T(S') \leq_2 T(S)$$

**Condition 2.** Model map and test case map preserve the pass relationship

$$\forall S \in Dom_1, U \in TC(Sig_2) : T(S) \text{ pass}_2 U \Leftrightarrow S \text{ pass}_1 T^*(U)$$
Satisfaction condition, reflected by commuting diagrams and relational composition

**Condition 1**

\[ T(Dom_1) \xrightarrow{\leq_2} T(Dom_1) \]

\[ Dom_1 \xrightarrow{\leq_1} Dom_1 \]

\[ T; \leq_2 = \leq_1; T \]

**Condition 2**

\[ T(Dom_1) \xrightarrow{\text{pass}_2} TC(Sig_2) \]

\[ Dom_1 \xrightarrow{\text{pass}_1} T^*(TC(Sig_2)) \]

\[ \text{pass}_1 = T; \text{pass}_2; T^* \]
Recall. Relational composition

\[ f \subseteq X \times Y, \quad g \subseteq Y \times Z \]

\[ f ; g = g \circ f = \{ (x, z) \mid \exists y : (x, y) \in f \land (y, z) \in g \} \]

**Condition 1**

\[ T(Dom_1) \xrightarrow{\leq 2} T(Dom_1) \]

\[ \leq_1 \]

\[ Dom_1 \xrightarrow{\leq_1} Dom_1 \]

\[ T; \leq_2 = \leq_1; T \]

**Condition 2**

\[ T(Dom_1) \xrightarrow{\text{pass}_2} TC(Sig_2) \]

\[ T \]

\[ Dom_1 \xrightarrow{\text{pass}_1} T^*(TC(Sig_2)) \]

\[ \text{pass}_1 = T; \text{pass}_2; T^* \]
General theorem for translation of testing theories

**Theorem 1.** Suppose \((T, T^*)\) exist and fulfil the satisfaction condition. Then every complete (sound, exhaustive) testing theory established in \(Sig_2\) induces a likewise complete (sound, exhaustive) testing theory on \(Sig_1\)

Proof of Theorem 1 by diagram chasing
Proof of Theorem 1 by diagram chasing

All reference models $M_2$ occurring in combination with fault domain $Dom_2$
Proof of Theorem 1 by diagram chasing

Maps each reference model $M_2$ of same fault model $(M_2, \leq_2, \text{Dom}_2)$ to its associated test suite.

$F_2(\text{Dom}_2) \to \text{Dom}_2 \to \leq_2 \to \text{TS}_2,\text{Dom}_2 \to \text{TS}_2(\text{Dom}_2)$

$F_1(\text{Dom}_1) \to \text{Dom}_1 \to \leq_1 \to \text{TS}_1,\text{Dom}_1 = T; \text{TS}_2,\text{Dom}_2; T^*$
Proof of Theorem 1
by diagram chasing

All reference models $M_1$ occurring in combination with fault model $(M_1, \leq_1, Dom_1)$, such that $T(M_1) \in F_2(Dom_2)$
Dom$_1$ is fixed: $T(Dom_1) \subseteq Dom_2$ holds
Proof of Theorem 1 by diagram chasing

Fulfils $\leq_1; T = T; \leq_2$

[Satisfaction condition, part 1]
Proof of Theorem 1 by diagram chasing

Fulfils \( \text{pass}_1 = T; \text{pass}_2; T^* \)
[Satisfaction condition, part 2]
Proof of Theorem 1
by diagram chasing

Fulfils $\text{pass}_2 = (\leq_2; TS_{2,Dom_2})$ iff theory is complete
Fulfils $(\leq_2; TS_{2,Dom_2}) \subseteq \text{pass}_2$ iff theory is sound
Fulfils $\text{pass}_2 \subseteq (\leq_2; TS_{2,Dom_2})$ iff theory is exhaustive
Proof of Theorem 1
by diagram chasing

Fulfils $\text{pass}_1 = (\leq_1; TS_{1,Dom_1})$ iff is $\text{pass}_2 = (\leq_2; TS_{2,Dom_2})$ [completeness]
Fulfils $\text{pass}_1 \subseteq (\leq_1; TS_{1,Dom_1})$ iff is $\text{pass}_2 \subseteq (\leq_2; TS_{2,Dom_2})$ [soundness]
Fulfils $\text{pass}_1 \subseteq (\leq_1; TS_{1,Dom_1})$ iff $\text{pass}_2 \subseteq (\leq_2; TS_{2,Dom_2})$ [exhaustiveness]

$\text{pass}_1 = (\leq_1; TS_{1,Dom_1})$$\subseteq$$\text{pass}_1$$\subseteq$$\text{pass}_2$$\subseteq$$\text{pass}_2$$\subseteq$$\text{pass}_2$$\subseteq$$\text{pass}_2$
Proof of Theorem 1 by diagram chasing

Let $\leq_1, \leq_2$ be relations,

$\text{Dom}_1 \xrightarrow{\text{pass}_1} \text{TS}_1(\text{Dom}_1)$

$\xrightarrow{\text{TS}_1 = T; \text{TS}_2; T^*}$

$\text{Dom}_2 \xrightarrow{\text{pass}_2} \text{TS}_2(\text{Dom}_2)$

$\xrightarrow{\text{TS}_2}$

$\text{F}_1(\text{Dom}_1)$

$\text{F}_2(\text{Dom}_2)$
Proof of Theorem 1 by diagram chasing
Proof of Theorem 1 by diagram chasing

\[ F_2(Dom_2) \xrightarrow{\leq_2} Dom_2 \xrightarrow{TS_2} TS_2(Dom_2) \]
\[ Dom_1 \xrightarrow{\leq_1} Dom_1 \xrightarrow{pass_1} TS_1(Dom_1) \]
\[ TS_1 = T; TS_2; T^* \]
Proof of Theorem 1 by diagram chasing
Proof of Theorem 1 by diagram chasing
Side remark. This shows how model-based testing tools for domains $Dom_1$ and $Dom_2$ should interact in a collaborative verification environment.
TTT application

- **Theorem 2.** Every complete (sound, exhaustive) FSM testing theory for
  - language equivalence or
  - language containment

induces a complete (sound, exhaustive) **equivalence class partition testing theory** with analogous conformance relations for Kripke structures with **infinite input domains**, bounded nondeterminism, and finite internal state and finite outputs.


TTT-application

• **Step 1. Transform the transition relation**

  ◆ Create a transition relation of the model

  ◆ Separate input variables, internal model variables, and output variables, by enumerating the latter

  ◆ Aggregate sequences of transitions between transient states into a single transition leading to a quiescent post-state
TTT-application

- Step 1. Transform the transition relation

- This leads to transition relation of the form

\[ \mathcal{R} \equiv \bigvee_{i \in \text{IDX}} \left( \alpha_i \wedge (m, y) = (d_i, e_i) \wedge (m', y') = (d_i, e_i) \right) \]

\[ \vee \bigvee_{(i, j) \in J} \left( g_{i, j} \wedge (m, y) = (d_i, e_i) \wedge (m', y') = (d_j, e_j) \right) \]

with

- Stability conditions \( \alpha_i \)
- Jump conditions \( g_{i,j} \)
- Only input variables occur free in \( \alpha_i, g_{i,j} \)
• **Step 2. Calculation of input equivalence classes**

  • Each satisfiable solution of

\[
\Phi_f \equiv \bigwedge_{i \in \text{IDX}} g_{i,f(i)} \text{ with } f : \text{IDX} \rightarrow \text{IDX} \text{ permutation}
\]

specifies one input equivalence class
TTT-application

- **Step 3. Creation of the model map**
  - Map Kripke model to minimal, observable FSM with

\[
\begin{align*}
\text{Input alphabet} & \quad \Sigma_I = \{ \Phi_f \mid \Phi_f \text{ is feasible}\} \\
\text{Output alphabet} & \quad \Sigma_O = \text{finite output domain of Kripke model} \\
\text{Internal states} & \quad Q = \{ q_i \mid i \in \text{IDX}\} \\
\text{Transition relation} & \quad h = \{ (q_i, \Phi_f, e_j, q_j) \mid f(i) = j\}
\end{align*}
\]
TTT-application

Step 4. Creation of the test case map

- FSM test cases are acyclic, terminating, single-input, output-complete FSMs
- FSM test cases interact with the FSM to be tested via language intersection as „parallel operator“
- FSM test inputs state-dependent value to SUT
- FSM test accepts SUT output and transits into next state with new input or into fail state
FSM test case
• **Step 4. Creation of the test case map**

  - Kripke structure test cases interact (this is one option) synchronously with the SUT

  - In contrast to FSM test cases, inputs to the SUT are strictly separated from monitoring of outputs
• **Step 4. Creation of the test case map**

• Consequently, one FSM test step leads to a more complex Kripke test step involving several transitions
• **Step 5. Proof of the satisfaction condition**

- The proof is independent on the selection of representatives from each equivalence class, whenever this class occurs as an input in an FSM test case.

- Consequently, the test strategy for Kripke structures can be combined with random selection of input data from each class.
Theory translation – model-theoretic underpinning

• Alternative A. Theory of Institutions

Test case map above corresponds to sentence translation map in theory of institutions – Need Grothendieck Institutions


• Semantics for CPS – time for a change of paradigm?

• **Multiple formalisms in CPS modelling**
  
  • Example 1. Testing theories and collaborative tool environments
  
  • **Example 2. Verification of emergent properties**

• Conclusions and future work

Multiple formalisms in CPS modelling – Example 2. Verification of emergent properties
Recall – train onboard speed control

Onboard main controller

Automated braking command

PLC brake controller

Train engine driver brake command

Emergency brake

Current speed

Current maximum speed

man_on, man_off

trigger, release

auto_on, auto_off
Verification of emergent properties

• Application scenario
  - Onboard controller has been verified and tested using SysML models with Kripke semantics
  - PLC has been verified and tested using FSM models

  • **Verification objective.** System satisfies emergent property
    
    **EP.** “As long as the speed is above emergency threshold, the emergency brakes stay active and cannot be manually released”

  • **Technical side condition.** EP shall be specified in CSP trace logic
Verification of emergent properties

- Problems to be solved
  - EP can only be specified by referring to properties of both the onboard main controller and the brake controller.
  - Properties related to brake controller are specified by FSM I/O sequences $x/y$ – e.g. via intersection with testing automaton.
  - Properties related to Onboard speed controller are specified by, e.g. **LTL formulas with shared I/O variables** as free symbols.
  - CSP trace logic formulas are specified over **traces of events and refusal sets**.
Verification of emergent properties

- **Observations**
  - FSM I/O-events x/y can be mapped to CSP channel events x.y
  - FSM parallel composition by intersection is similar to synchronous channel communication of CSP processes
  - CSP failures models can be represented by normalised transition graphs

Alternative approach

- **Alternative B. Approach based on Unifying Theories of Programming UTP**

  - „Programs are predicates“ – no distinction between models and sentences
  
  - Theories are made up from alphabets, signatures, and healthiness conditions
  
  - Conformance is expressed by implication $[P \Rightarrow Q]$ („$P$ refines $Q$“)
  
  - Model, sentence, and theory translation is enabled by the existence of Galois connections

---

Verification of emergent properties

• **Procedure**

  • Create UTP theories for

    ◆ Sub-class of Kripke structures (sequential nondeterministic programs) with LTL safety formulas for property specifications,

    ◆ FSMs with property specification by testing automata

    ◆ CSP failures model with failures (= trace/refusal) specifications
      \[ P \text{ sat } S(tr, ref) \]

    ◆ CSP transition graphs with CSP-like specifications
      \[ G \text{ sat } S(tr, ref) \]
Verification of emergent properties

**Procedure**

- Create Galois connections
  - CSP failures models $\Leftrightarrow$ CSP transition graphs
  - Sequential nondeterministic programs $\Leftrightarrow$ CSP transition graphs
  - FSMs $\Leftrightarrow$ CSP transition graphs
- This allows us to
  - lift the local properties of FSM and Kripke structure to local CSP assertions
  - deduce the required satisfaction relation on CSP level by means of compositional reasoning
Theory translation – model-theoretic underpinning

Application example

• Some of these Galois connections have already been established

Ana Cavalcanti, Wen-ling Huang, Jan Peleska, Jim Woodcock: CSP and Kripke Structures. ICTAC 2015: 505-523
• Semantics for CPS – time for a change of paradigm?

• Multiple formalisms in CPS modelling
  • Example 1. Testing theories and collaborative tool environments
  • Example 2. Verification of emergent properties

• Conclusions and future work

Conclusions and future work
Conclusions

• We have identified characteristics of CPS challenging the existing semantic approaches to concurrent systems

• Potential solutions to the problems of
  ♦ theory translation
  ♦ verification of emergent properties in presence of multiple formalisms

have been proposed
Future work

• Evolution of asserted behaviour
  ◆ Inspiration from AI. **Belief systems** and belief revision – CPS components should act optimally in relation to the current status of belief – belief revision should only be necessary within specified boundaries

• Semantic navigation
  ◆ A network of semantics offering different degrees of abstraction
  ◆ Network nodes are connected by **theory translation mappings** – Galois Connections?

• Dynamic re-configuration
  ◆ Simpler methods are available for **bounded-length model investigation**, as used in bounded model checking and model-based testing
Acknowledgements

I would like to express my gratitude to this audience, its organisers, and to my friends and collaborators who inspired and contributed to the ideas presented in this talk.

Ana Cavalcanti, Anne E. Haxthausen, Wen-ling Huang, Christoph Hilken, Felix Hübner, John Fitzgerald, Peter Gorm Larsen, Till Mossakowski, Mohammad Reza Mousavi, Alexandre Petrenko, Markus Roggenbach, Uwe Schulze, Linh Hong Vu, Jim Woodcock, Cornelia Zahlten

The work presented here has been performed in the context of project Implementable Testing Theories for Cyber-physical systems (ITTCPS) http://www.informatik.uni-bremen.de/agbs/projects/ittcps/index.html