Towards Verifying Cyber-Physical Systems with Structural Dynamism
Dagstuhl Seminar 11441-1 Science and Engineering of Cyber-Physical Systems, 02.11.2011

Holger Giese and Basil Becker
System Analysis & Modeling Group, Hasso Plattner Institute for Software Systems Engineering at the University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
Application Example: Combine shuttles as a CPS ...

http://www.railcab.de/

Test shuttle

Test track

A shuttle system that builds convoys to optimize the energy consumption
1) Modeling with Graph Transformation Systems

Apply Graph Transformation Systems

- Map the tracks
- Map the shuttles
- Map the shuttle movement to rules (movement equals reconfiguration)

Rule:

- t1:Track → t2:Track
- t1:Track → t1:Track
- s1:Shuttle → s1:Shuttle
2) Modeling with Graph Transformation Systems

**Forbidden Graph**

- **Correctness**: all reachable system graphs do not match the forbidden graph pattern

**Idea for hybrid behavior**: continuous attributes and modes with continuous laws

- **Correctness**: all reachable hybrid system graphs do not match the forbidden hybrid graph pattern

---

02.11.2011 | Giese & Becker | Towards Verifying CPS with Structural Dynamism
Modeling the Railcab System

**Meta Model:**

- **SpeedControlMode**
  - \( v_{ref} : \text{float} \)
- **PositionControlMode**
  - \( pos_{ref} : \text{float} \)
  - \( v_{ref} : \text{float} \)

**ControlMode**

- \( \text{pos} - \text{rear.front.pos} \leq \delta \)

**Continuous Behavior:**

- **SpeedControlMode**
  - \( v_{ref} : \text{float} \)
- **PositionControlMode**
  - \( pos_{ref} : \text{float} \)
  - \( v_{ref} : \text{float} \)

- \( \dot{p}_s = v \)
- \( \dot{v} = a \)
- \( \ddot{a} = v_{ref} - v \)
- \( \dot{v} = \frac{(pos_{ref} - pos)}{10^5(v_{ref} - v)} \)

**Discrete Behavior (Rule):**

- \( cm : \text{PositionControlMode} \)
- \( dc : \text{DistanceCoordination} \)
- \( s1 : \text{Shuttle} \)
- \( s2 : \text{Shuttle} \)
- \( t1 : \text{Track} \)
- \( t2 : \text{Track} \)

**Forbidden Situation (Graph Pattern):**

- \( noDC : \text{DistanceCoordination} \)
- \( s1 : \text{Shuttle} \)
- \( s2 : \text{Shuttle} \)
- \( t1 : \text{Track} \)
- \( t2 : \text{Track} \)
Basic Verification Idea

Idea (invariant checking):
- Look only for a transition from a safe to an unsafe state
- Found a case leading from a safe to a forbidden graph pattern

Timed:
- Found a case leading from a safe to a forbidden graph pattern also fulfilling the time constraints that is not prevented by other rules (system of linear inequality; CPLEX solver)

Hybrid:
- Construct hybrid automata for the check (PHAVer)
Verification of the Application Example

- Structural Check returns possible counterexamples (not taking the continuous behavior and constraints into account)

- Modelchecking a related hybrid automata disproof or conforms each counterexample

```
automaton GenericHybridGTS
contr_var: s1.pos, s2.pos, s1.v, s2.v, s1.a, s2.a, v.ref, pos.ref, timer;
parameter: distance, failure;
synch: void;
loc sourcePattern: while t<0 wait {timer' := 1};

when timer >= 0 sync void do {pos.ref' := s1.pos - distance - 2 & s1.pos' := s1.pos & s2.pos' := s2.pos & s2.v' := s2.v & s1.v' := s1.v & s1.a' := s1.a & s2.a' := s2.a & v.ref' := v.ref & failure' := 0} goto targetPattern;
loc targetPattern: while s1.pos - distance - s2.pos >= 0 wait {s1.pos' := s1.v & s1.a' := s1.a & s2.a' := P + (v.ref - s1.v & s2.v' := P.2 + (s1.pos - distance - 10 - s2.pos) - Q.2 + (s2.v - 3 - s1.v) & pos.ref' := s1.pos' & v.ref' := 0};
when s1.pos - distance - s2.pos <= 0 sync void do {failure' := 1} goto failureState;
loc failureState: while true wait {true};
loc urgentTransition: while true wait {true};
initially: sourcePattern & s1.pos > s2.pos + distance + 10 & s2-pos > 0 & 60 < v-ref & v-ref < 200 & 60 < s1.v & s1.v < 200 & 3 <= s1.v - s2.v & s1.v - s2.v <= 3 & failure' := 0 & 5 < distance & distance < 10;
```
Summary

- Very *expressive* model in form of hybrid graph transformation model containing
  - **Discrete behavior** with structural dynamism (which potentially leads to a discrete infinite states paces in form of graphs)
  - **Continuous behavior** in form of mode nodes and their continuous laws that can in principle reference all continuous variables of reachable other nodes
- Invariant checker for restricted variant where for all counter-examples a closed continuous system of inequalities can be derived.
- Tool support is still under development ...