Hasso Plattner Institut

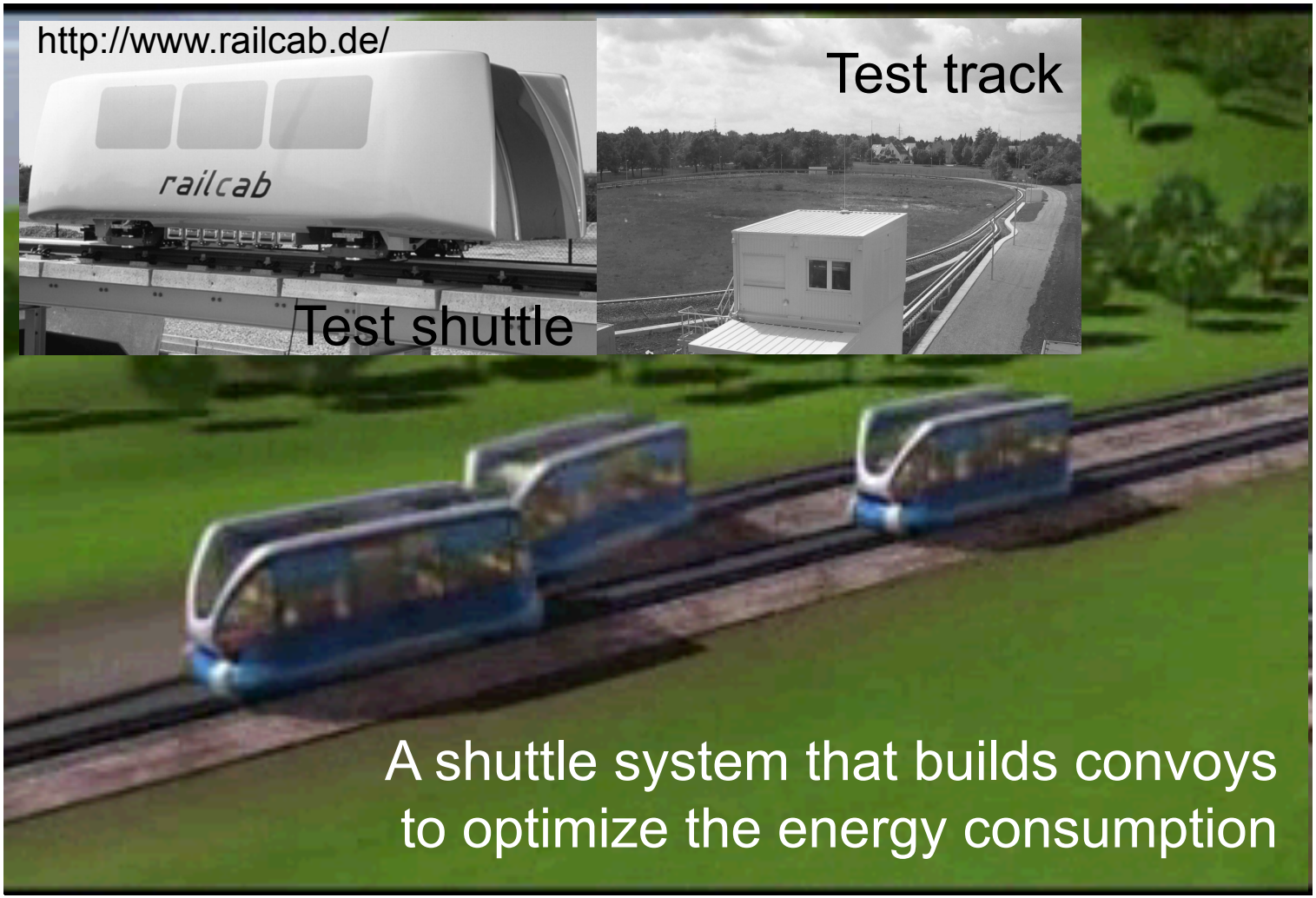
IT Systems Engineering | Universität Potsdam

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

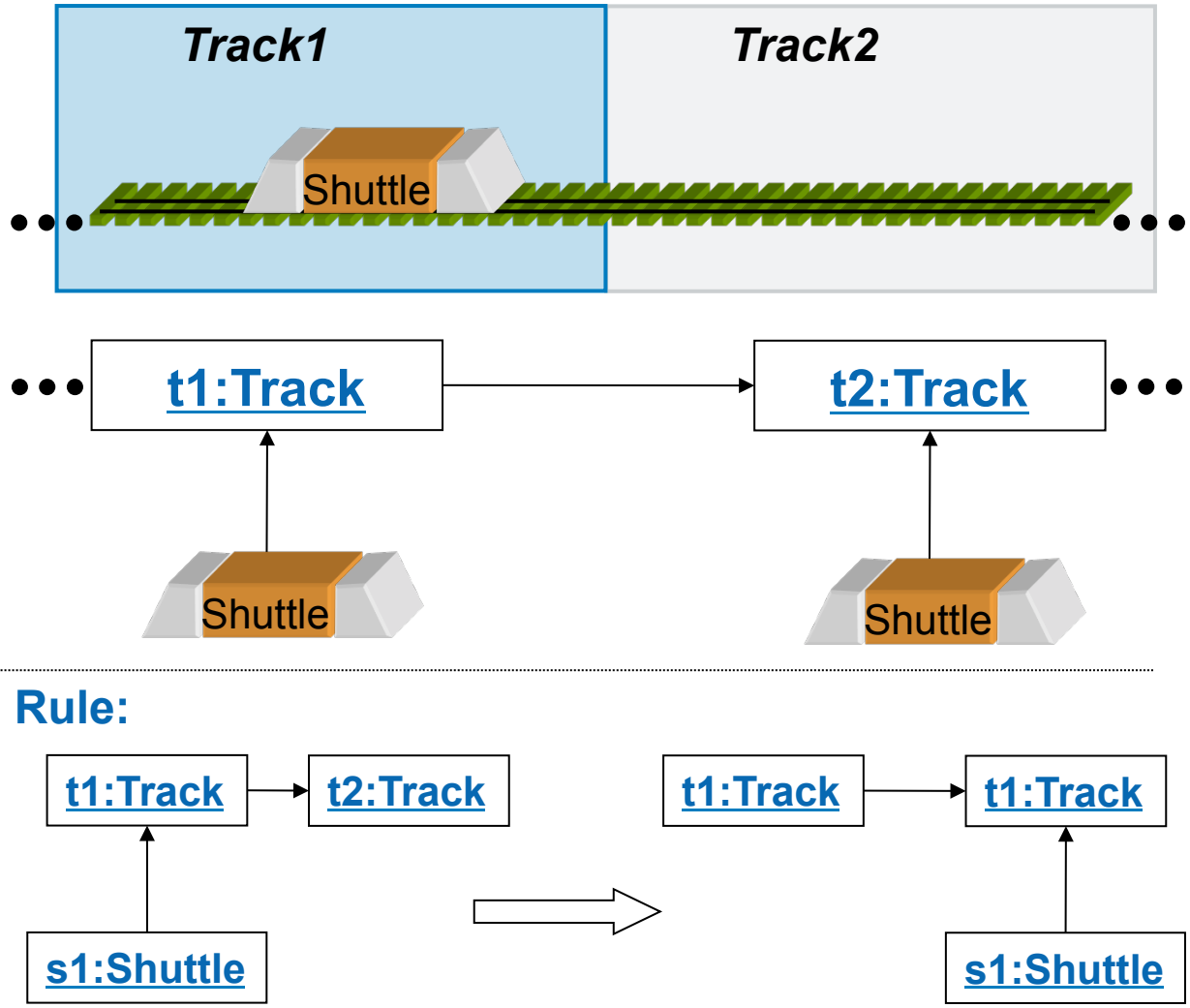
Application Example:
Combine shuttles as a CPS …



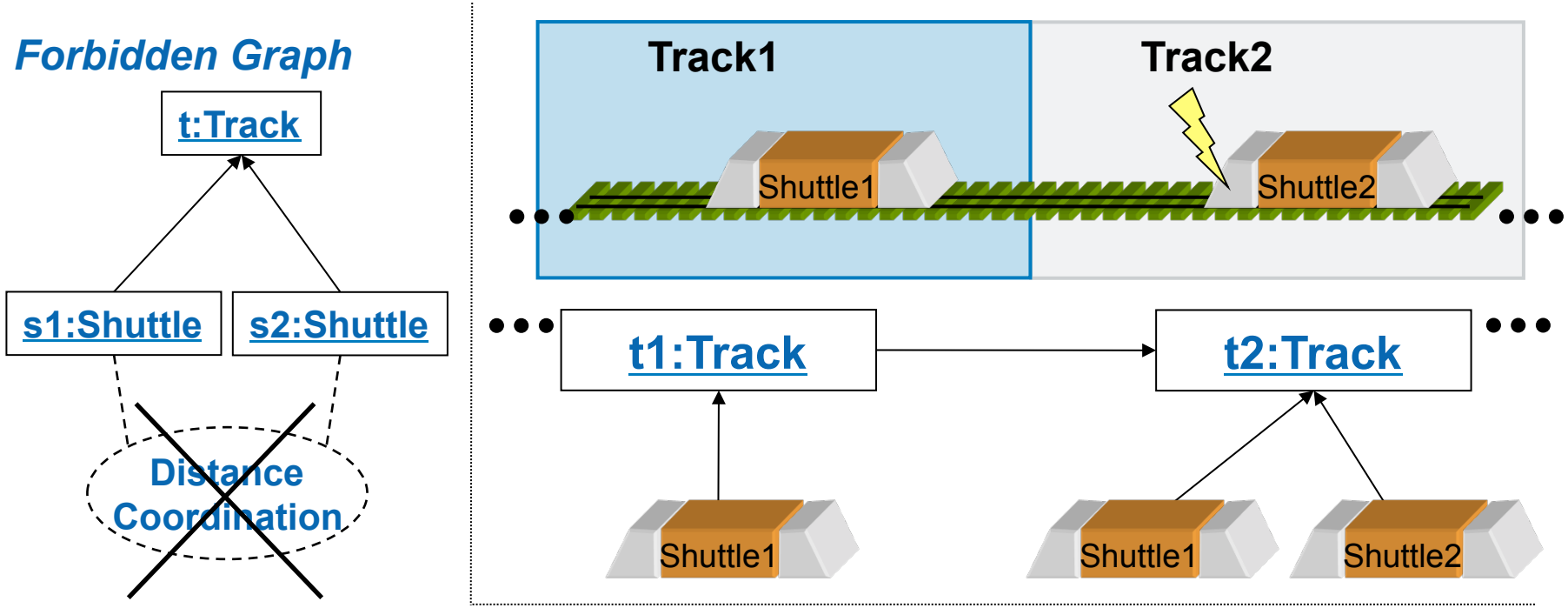
1) Modeling with Graph Transformation Systems

Apply Graph Transformation Systems

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



2) Modeling with Graph Transformation Systems



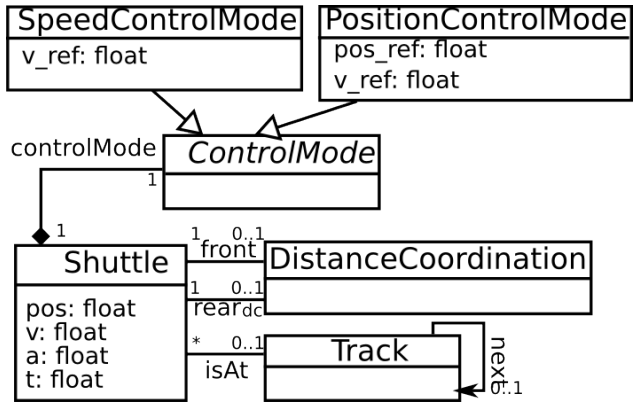
- **Correctness:** all reachable system graphs do not match the forbidden graph pattern

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

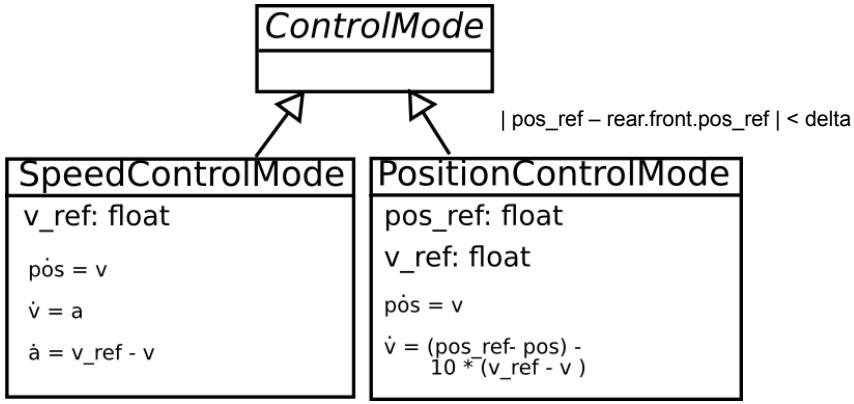
- **Correctness:** all reachable hybrid system graphs do not match the forbidden hybrid graph pattern

Modeling the Railcab System

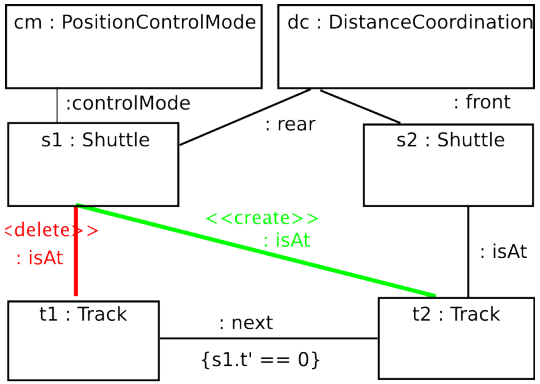
Meta Model:



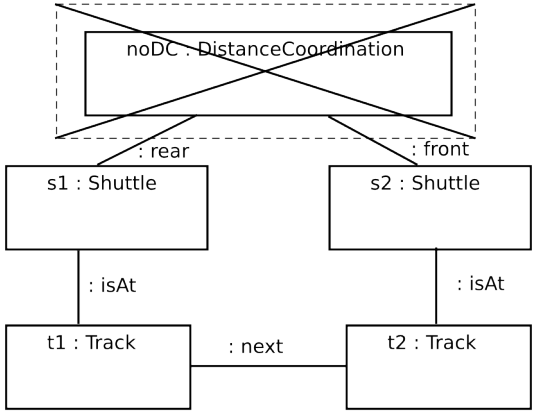
Continuous Behavior:



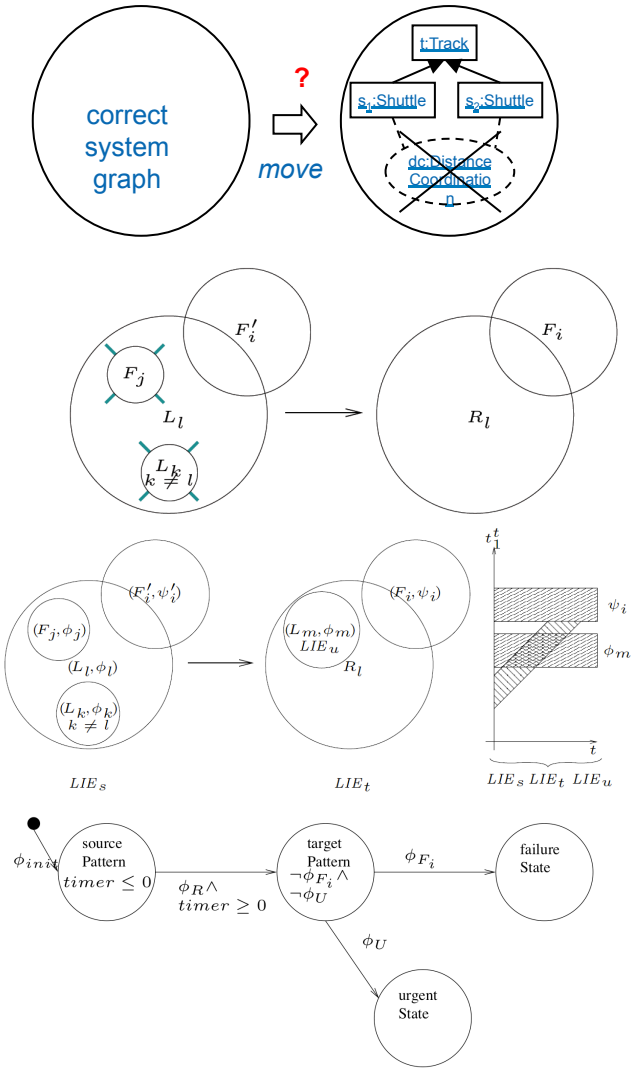
Discrete Behavior (Rule):



Forbidden Situation (Graph Pattern):



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:

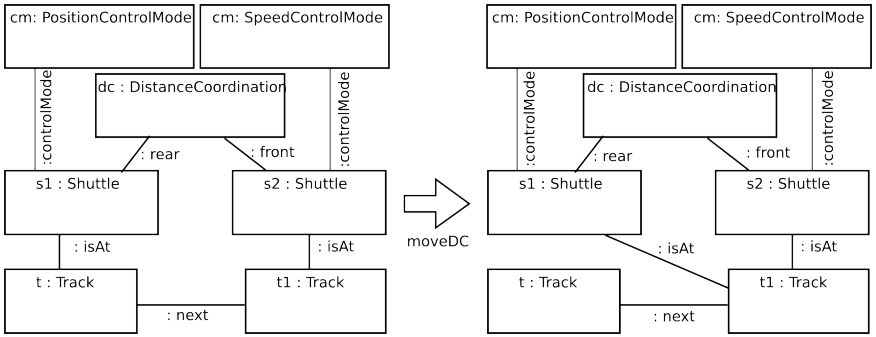
- Found a case leading from a safe to a forbidden graph pattern also fulfilling the time constraints that is not prevented by other rules (system of linear inequality; CPLEX solver)

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;
  synclabs: 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_v'
       == s1_a & s1_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;
end
```

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 …