Basic problems in multi-view modeling

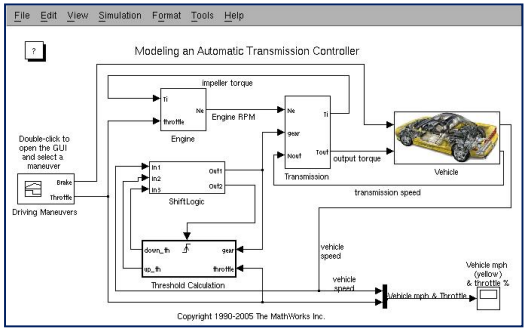
Stavros Tripakis (UC Berkeley)

ExCAPE meeting, March 2014

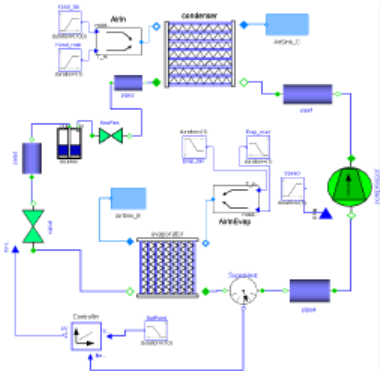
Joint work with Jan Reineke
(Saarland University)

Multi-View Modeling

Complex system -> many design teams -> many viewpoints
-> many perspectives -> many models = views

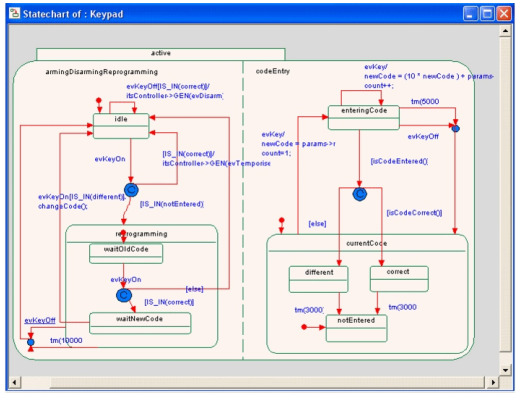


Low-level controllers Simulink



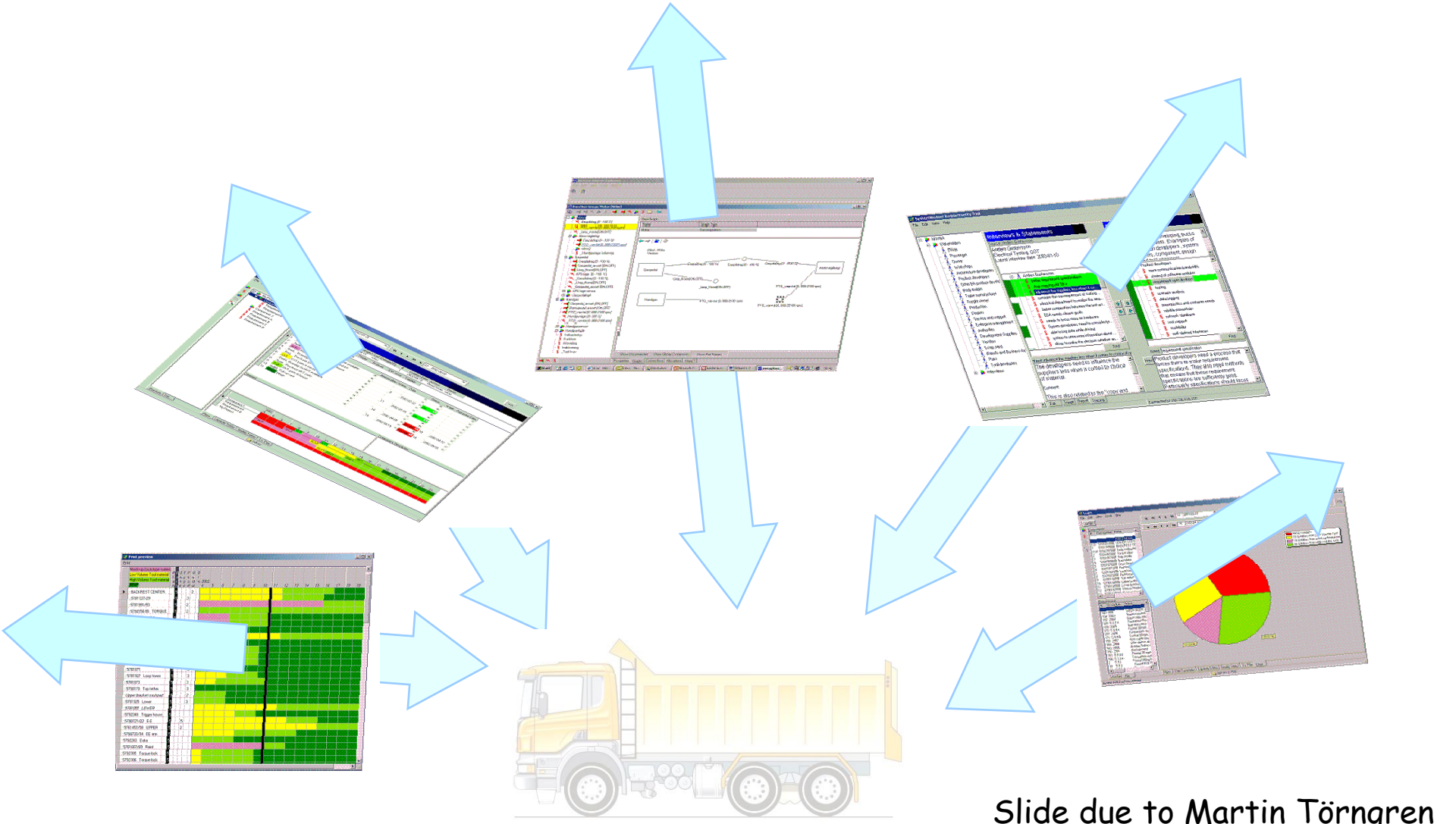
Physical dynamics Modelica

Supervisory controllers Rhapsody/ SysML



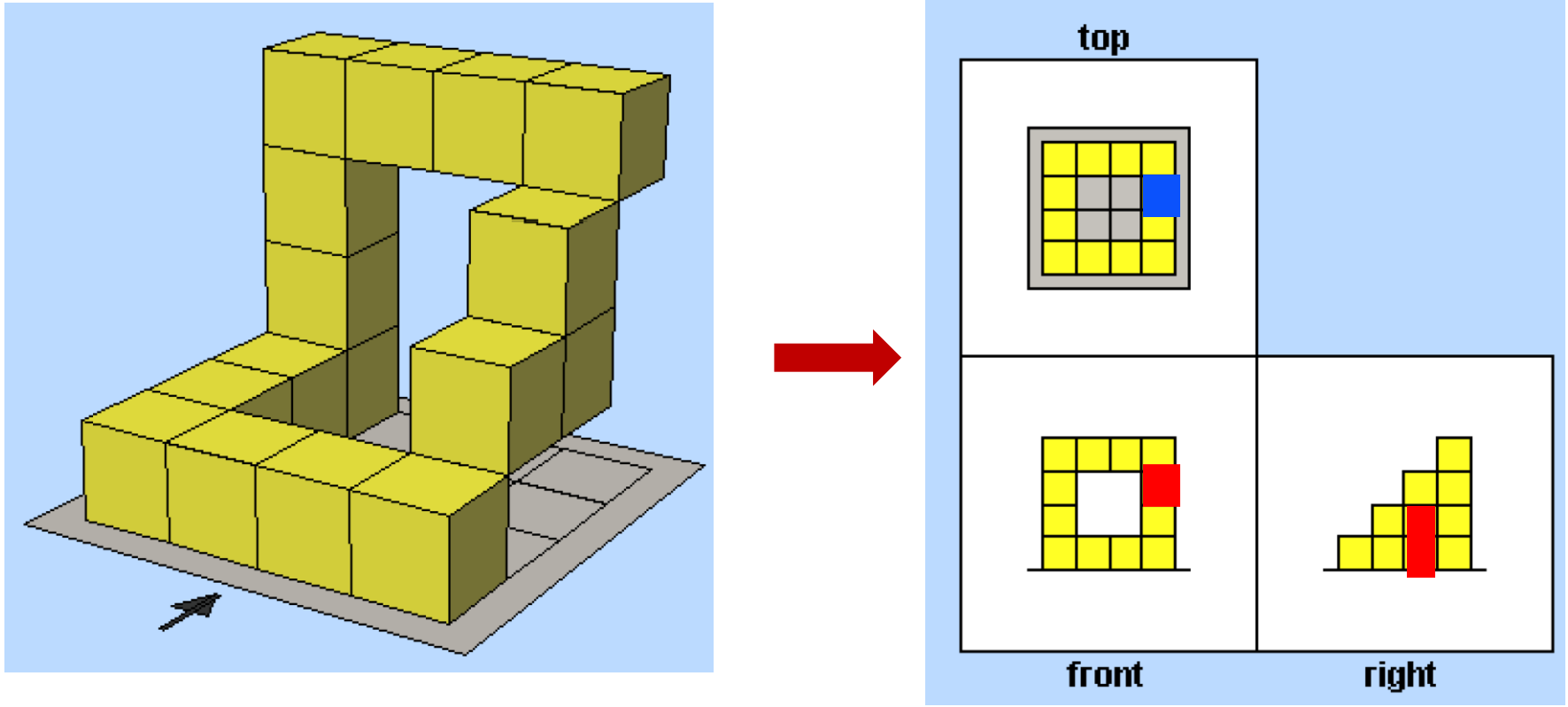
Problem: view consistency

Partially overlapping content -> potential for contradictions



Slide due to Martin Törngren

Example: geometric (static) views

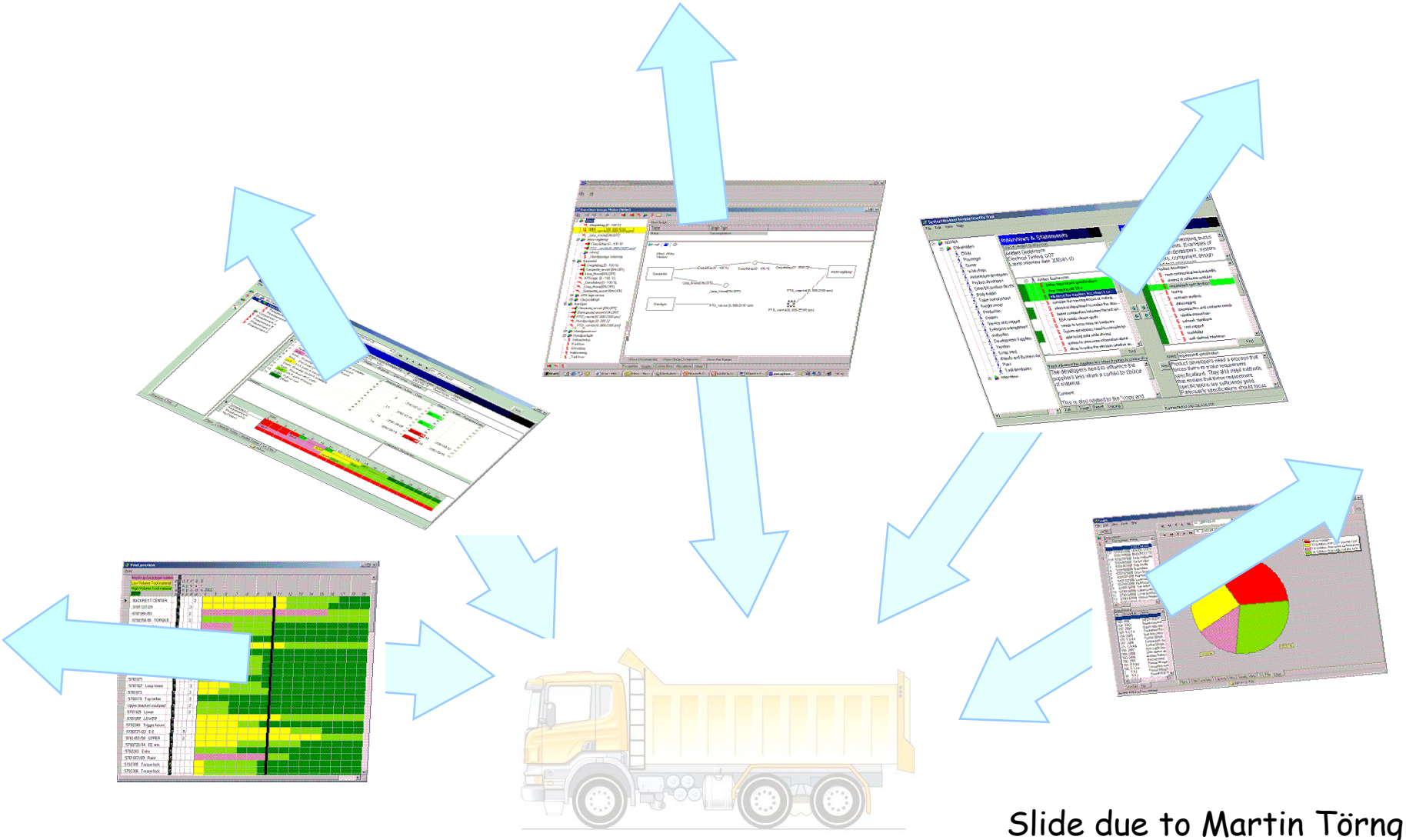


3D structure

2D views

Goals: verification and synthesis

Want to: (1) check consistency, (2) synthesize system



Slide due to Martin Törngren

This work [TACAS 2014]

- Study **behavioral** (dynamic) views
- Part 1: abstract formal view framework
- Part 2:
  - Instantiate the framework for discrete systems
  - Study different verification and synthesis problems (decidability, complexity, …)

So what are views, formally?

- Semantically, systems and views are sets of behaviors
- Syntactically, they can be any formalism that generates behaviors (e.g., automata, symbolic transition systems, …)

- Views are **generated** from systems
  - View = "aspect" of a system

How are views generated from systems?

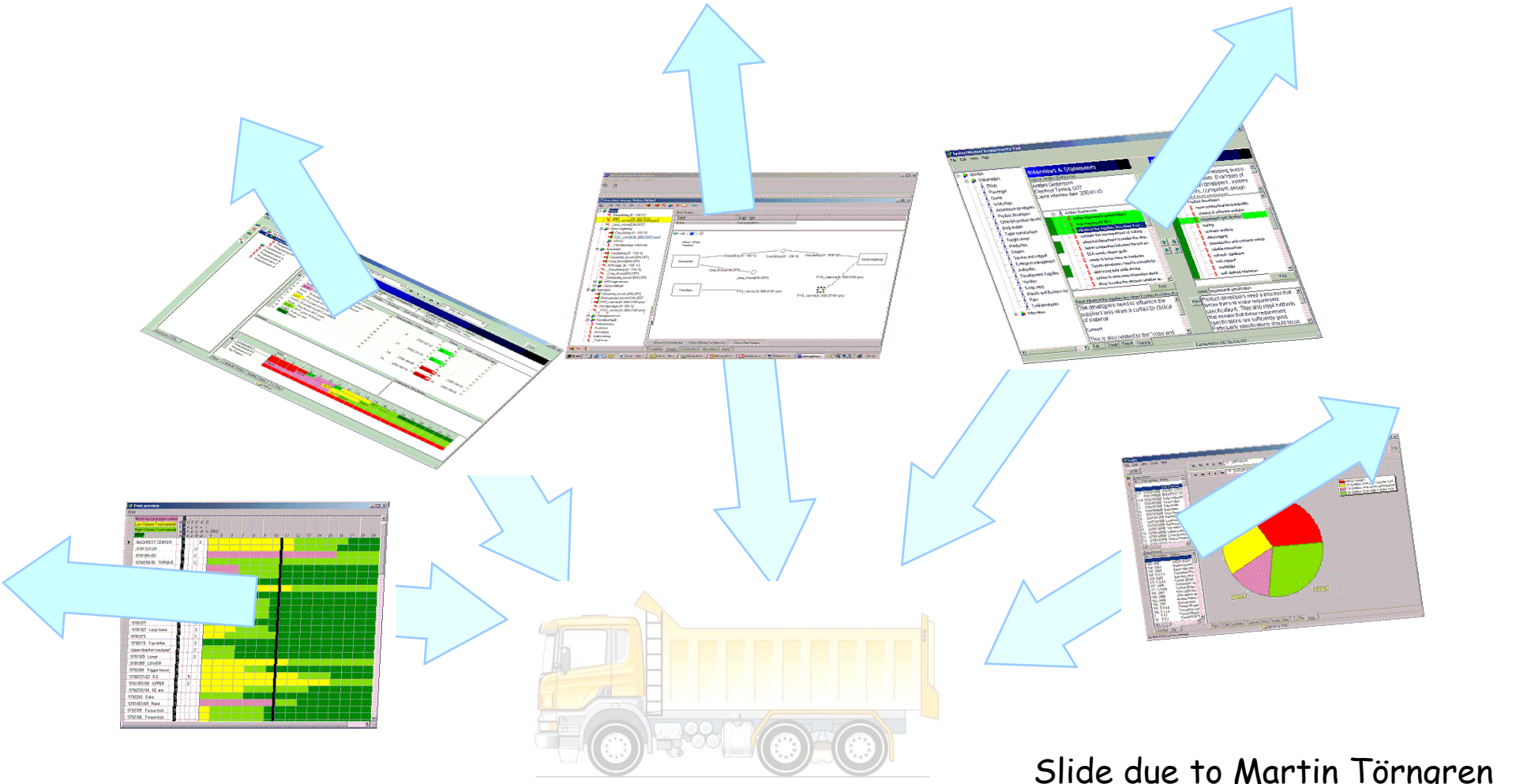
- Intuition: a view of a system is a **projection** of it
- More generally: views are defined by **abstraction functions**

- E.g.,
  - A system has 5 state variables x,y,z,a,b
  - One view of the system is its projection to the 3 variables x,y,z
  - Another view is the projection to x,a,b

What is view consistency?

A set of views are consistent =

∃ system that could generate these views



Slide due to Martin Törngren

What is view consistency?

- Views V1 and V2 are consistent w.r.t. abstraction functions a1 and a2 iff ∃ a **witness** system S s.t. V1=a1(S) and V2=a2(S).

- Note: this is different from checking satisfiability of the conjunction of a set of, say, temporal logic formulas $\varphi 1 \wedge \varphi 2$.
  - Here we require =, not ∈

Views vs. component contracts

- Component contracts [c.f. Rohit's and Antonio's presentations]:
  - Refer to clearly separated system **components**
  - Components interact via inputs-outputs, but typically don't overlap
- Views can be seen as contracts, too:
  - But views talk about different system **aspects**, not components
  - Views typically overlap

Some verification and synthesis problems on views

- View **consistency checking**: given views V1, …, Vn, check whether they are consistent w.r.t. abstraction functions a1, …, an

- System **synthesis** from views: if they are consistent, synthesize witness system
  - A.k.a. "Model-Based Design"

Some results

- View consistency checking is decidable (PSPACE-complete) for discrete systems
  - Discrete systems = symbolic transition systems (possibly **distinguishing observable and unobservable state variables**)
  - Abstraction functions = projections to subsets of state variables

More in our TACAS paper

- "Basic Problems in Multi-View Modeling", by Jan Reineke and Stavros Tripakis, in Tools and Algorithms for the Construction and Analysis of Systems (TACAS) 2014

Lots more to be done

- Extend to input-output systems and address potential game-theoretic issues that may arise ($\forall - \exists$).
- Heterogeneous views.

Back-up slides

View conformance

- Sometimes V = a(S) is too strict.
- We may instead want V to be subset/superset of a(S).

View reduction

- Knowing something about V2, V3, …, we can deduce information about V1.

Orthogonality

- Definition 1: view domains D1 and D2 are orthogonal iff any pair of non-empty views chosen from them are consistent w.r.t. =.
  - Imagine the axes X, Y, Z, but non-empty sets of points.
- Definition 2: (non-empty) views V1, V2, …, Vn, are orthogonal iff they are mutually irreducible.
- The two definitions are equivalent.

Verification and synthesis problems on views

- View synthesis: given system S and abstraction function a, synthesize a(S)
- View consistency checking: given V1, …, Vn, check consistency.
- System synthesis from views: given V1, …, Vn, check consistency, and if consistency, synthesize witness system (perhaps with some "optimality" criteria).
- View reduction: given V1, …, Vn, synthesize V1',….,Vn' which are irreducible.

Instantiation

- Apply this abstract framework to discrete systems
- Discrete system: Boolean variables + non-deterministic dynamics (transition relation)
- Abstraction functions: projections ("hide" some variables)

View synthesis

- Simplest problem: given discrete system, want to hide some of its variables.
- Fact: **fully-observable** discrete systems not closed under projection.
- So: consider systems with both **observable** and **unobservable** variables.

Remaining paper

- Study the various view-related problems on discrete systems with observable and unobservable variables.

- E.g.,
- "Consistency is PSPACE-complete for =".

Note

- View consistency vs. satisfiability of conjunction of LTL formulas:
  - Suppose views are LTL formulas: f1, f2, …, fn.
  - These could refer to different sets of variables.
  - Does view consistency just mean that conjunction F = f1 & f2 & … & fn should be satisfiable?
- No:
  - F sat means there exists one behavior **satisfying** (i.e., element of) all views.
  - Consistency (w.r.t. =) means there exists system S such that the i-th projection of S is **equal to** fi.