A formal framework for multi-view modeling and the multi-view consistency problem

Maria Pittou
(joint work with Prof. Stavros Tripakis)

Aristotle University, School of Sciences, Department of Mathematics

Seminar on TCS, 19/12/2018
Motivation
System modeling

- Powerful technique for reasoning about systems
- Provides abstractions of the system under development
- Detects errors and improves performance
Motivation
System modeling formalisms

- State machines and automata

- Differential equations $\rightarrow$ physical plant of CPSs

- Timed and Hybrid Automata

Motivation

Complex systems
Motivation
Modeling of large and heterogeneous systems

The construction of any large and complex system involves multiple design teams with their own perspectives of the system.
The construction of any large and complex system involves multiple design teams with their own view of the system.

In multi-view modeling (MVM for short) the different stakeholders derive separate but yet related models, called views, of the same system.
One of the main challenges in multi-view modeling is to ensure consistency among the different views.
Outline

1 Motivation for multi-view modeling

2 Related work

3 Formal framework for multi-view modeling

4 Contributions to the formal framework

5 Generic algorithm for checking view consistency

6 Challenges in MVM and future work
Related work

- Early work on multi-view modeling
- Multi-modeling languages
- Multi-view modeling for embedded and cyber-physical systems
- Other approaches to multi-view modeling
- A formal framework for multi-view modeling
Related work
Early work on MVM

  - One of the first papers that studies the problem
  - Informal framework illustrated by a lift system
  - Tool support discussion
Related work
Early work on MVM


- A reference model is used to generate the views

- View inconsistencies and integration

- Specification by graph transformations and illustration by a banking system
Related work

Early work on MVM

- Living with inconsistency in large systems, 1988, R. W. Schwanke and G. E. Kaiser
  - Consistency with respect to type safety in programming languages
  - Inconsistency is sometimes unavoidable and more effective
  - CONMAN programming environment
Multi-view modeling is supported by multi-modeling languages.

Multi-modeling languages provide diagrams to specify abstractions of software and hardware systems.

UML and SysML are multi-modeling languages.
Related work

Multi-modeling languages

- Semantically configurable consistency analysis for class and object diagrams, 2011, S. Maoz et al.

- Views of a system are described by class and object diagrams
Related work
Multi-modeling languages

- Semantically configurable consistency analysis for class and object diagrams, 2011, S. Maoz et al.

- Views of a system are described by class and object diagrams

- Semantic consistencies are investigated

- Automated consistency checking
Related work

Multi-modeling languages

- **Vooduu**: Verification of Object-Oriented Designs Using UPPAAL, 2004, K. Diethers and M. Huhn
  - Views are state or sequence diagrams with timing constraints
  - Multi-view consistency is reduced to model checking
  - Tool that enables verification of UML diagrams

- View relations for describing multi-view systems

- Main characteristics of views and basic challenges in multi-view modelling

- A nice survey on different approaches to MVM
Related work
Multi-view modeling for embedded and cyber-physical systems


Content relationships between views.

- View relations for describing multi-view systems
- Main characteristics of views and basic challenges in multi-view modelling
- A nice survey on different approaches to MVM
Related work

Other approaches to multi-view modeling

- Metamodeling
- Aspect-Oriented Modeling
- Interface Theories
Related work

Formal framework for multi-view modeling

A formal framework for multi-view modeling

Related work
Methods on MVM

- Structural views
- Behavioral views
- Consistency
  - Conjunctive approach
  - Non-conjunctive approach

Handling inconsistencies
Related work

Methods on MVM

- Structural views
- Behavioral views

Consistency

- Conjuctive approach
- Non-conjuctive approach

Handling inconsistencies
Outline

1 Motivation for multi-view modeling
2 Related work
3 Formal framework for multi-view modeling
4 Contributions to the formal framework
5 Generic algorithm for checking view consistency
6 Challenges in MVM and future work
Problem to be solved

Given a (finite) set of views, are they consistent?

1) How are the views (and the system) defined?

2) How are the views derived from the system?

3) What does view consistency mean?

4) How do we synthesize a system from its views?
Systems and views are defined \textit{semantically}.

- System $S$: set of behaviors
- View $V$: set of behaviors
- Abstraction function $V = a(S)$
Views are intuitively an **incomplete** picture of a system.

- Some behaviors may be missing from the view
- Some parts of a behavior itself may be missing in the view.
- A view may be obtained by some other kind of transformation

**Examples from:** Basic problems in multi-view modeling, 2014, J. Reineke and S. Tripakis.
Formal framework for multi-view modeling
System, views, and abstraction functions notations

- $U$ is the set of all possible system behaviors
- $S \subseteq U$ is a system
- $D_i$ denotes the set of view behaviors
- $V_i \subseteq D_i$ is a view
- $a_i : U \rightarrow D_i$ is an abstraction function
Formal framework for multi-view modeling

The multi-view consistency problem

View 1
V_1

... 

View n
V_n
Formal framework for multi-view modeling

The multi-view consistency problem

? S

Consistency ?

View 1
V₁

...  

View n
Vₙ
Formal framework for multi-view modeling

The multi-view consistency problem

![Diagram]

- $S$
- $a_1(S)$
- $a_n(S)$
- Consistency
- View 1
  - $V_1$
- View $n$
  - $V_n$
Formal framework for multi-view modeling

The multi-view consistency problem

View consistency

The views $V_1, \ldots, V_n$ over view domains $D_1, \ldots, D_n$ are consistent with respect to the abstraction functions $a_1, \ldots, a_n$, if there exists a system $S$ over $U$ so that $V_1 = a_1(S), \ldots, V_n = a_n(S)$. 

We call such a system $S$ a witness system to the consistency of $V_1, \ldots, V_n$. If there is no such system, then we say that the views are inconsistent.
Formal framework for multi-view modeling
The multi-view consistency problem

View consistency

The views $V_1, \ldots, V_n$ over view domains $D_1, \ldots, D_n$ are consistent with respect to the abstraction functions $a_1, \ldots, a_n$, if there exists a system $S$ over $U$ so that $V_1 = a_1(S), \ldots, V_n = a_n(S)$.

- We call such a system $S$ a witness system to the consistency of $V_1, \ldots, V_n$.
- If there is no such system, then we say that the views are inconsistent.
Consistency as a special case of conformance

- Consistency is described by strict equality $V = a(S)$.
- Consistency is a special case of conformance.
- **Conformance** expresses how faithful is a view to a system.
Consistency as a special case of conformance

- **Conformance** expresses how faithful is a view to a system.

- Conformance is defined w.r.t. a partial order \( \sqsubseteq \) on the powerset of a view domain \( D \).

- A view **conforms** to a system w.r.t. \( a \) iff \( V \sqsubseteq a(S) \).

- Consistency \( V = a(S) \) is a special case of conformance.
1 Motivation for multi-view modeling

2 Related work

3 Formal framework for multi-view modeling

4 Contributions to the formal framework

5 Generic algorithm for checking view consistency

6 Challenges in MVM and future work

→ **System, Views**: symbolic discrete systems (transition systems).

→ **Abstraction functions**: variable hidings.

![Diagram](attachment:diagram.png)
Contributions to the formal framework

- **Basic problems in multi-view modeling, 2014.**
  - **System, Views:** symbolic discrete systems (transition systems).
  - **Abstraction functions:** variable hidings.

![Diagram showing transitions between states](image)
Contributions to the formal framework

  - System, Views: symbolic discrete systems (transition systems).
  - Abstraction functions: variable hidings.

```
(0, 0, 0) → (0, 0, 1)
(1, 1, 1) ← (0, 1, 0)
(0, 0) → (0, 1)
(1, 1) ← (1, 0)
```
Contributions to the formal framework

  - **System, Views**: either finite automata or $\omega$-finite automata
  - **Abstraction functions**: projections of an alphabet of events onto a subalphabet.

![Diagram of automata]
Contributions to the formal framework

- Checking multi-view consistency of discrete systems with respect to periodic sampling abstractions, 2018.

  → **System, Views**: symbolic transition systems or Buchi automata

  → **Abstraction functions**: timing abstractions (periodic samplings)

  ⇔ **A necessary and sufficient** condition for **view consistency independent** of the particular **instantiation** of systems, views, and abstraction functions
Buchi automata and symbolic transition systems are prominent modeling structures.

Both are used by widespread verification tools such as SMV or Spin.

Sampling (time-driven/periodic, or event-driven) is a widely used mechanism in observation, control, embedded software etc.

→ example: drive-by-wire system in a modern car (sensors are periodically sampling some physical values)
1. Motivation for multi-view modeling
2. Related work
3. Formal framework for multi-view modeling
4. Contributions to the formal framework
5. Generic algorithm for checking view consistency
6. Challenges in MVM and future work
Canonical witness system candidate

- $V_1, \ldots, V_n$ are views
- $a_1, \ldots, a_n$ respective abstraction functions

\[
S = \bigcap_{i=1}^n a_i^{-1}(V_i)
\]

**Theorem**

$V_1, \ldots, V_n$ are consistent iff for all $i = 1, \ldots, n$ it holds that $a_i(S) = V_i$.

⇒ The canonical witness $S$ is the most general witness.
Generic algorithm for checking view consistency

Canonical witness system candidate

- $V_1, \ldots, V_n$ are views
- $a_1, \ldots, a_n$ respective abstraction functions

\[ S = \bigcap_{i=1}^{n} a_i^{-1}(V_i) \]

Algorithm to check view consistency

1. Compute $S$, i.e., compute the inverse abstractions $a_i^{-1}(V_i)$ and their intersection.
2. Compute $a_i(S)$ and check whether $a_i(S) = V_i$. 
Instantiating the generic algorithm

Algorithm to check view consistency

1. Compute \( S = \bigcap_{i=1}^{n} a_i^{-1}(V_i) \)
2. Compute \( a_i(S) \) and check whether \( a_i(S) = V_i \).

Checking consistency of two Büchi automata views

- \( V_1 \) and \( V_2 \) are nondeterministic Büchi automata (NBA).
- \( a_1^{-1} \) and \( a_2^{-1} \) are inverse periodic samplings.
- \( S = a_1^{-1}(V_1) \cap a_2^{-1}(V_2) \) is NBA intersection.
- \( a_1(S) \) and \( a_2(S) \) are periodic samplings.
- \( a_1(S) = V_1 \) and \( a_2(S) = V_2 \) is an NBA language equivalence problem.
Checking consistency of discrete systems w.r.t. periodic sampling

Overview of results

- Checking multi-view consistency of discrete systems with respect to periodic sampling abstractions, 2018

- A necessary and sufficient condition for checking view consistency.

- Solution for the multi-view consistency problem for views described by NBA.

- Solution for the multi-view consistency problem for views described by symbolic transition systems.

- Complexity analyses for the various algorithms.
Outline

1. Motivation for multi-view modeling
2. Related work
3. Formal framework for multi-view modeling
4. Contributions to the formal framework
5. Generic algorithm for checking view consistency
6. Challenges in MVM and future work
Main challenges in MVM

- View consistency
- Conformance checking
- View reduction
- Orthogonality of views
- Extendability of views
- Automation, view reuse, change propagation...
Future work
For the formal framework of MVM

- Heterogeneous instantiations of the formal MVM framework.
- Real case study for the formal MVM framework.

*Thank you!*