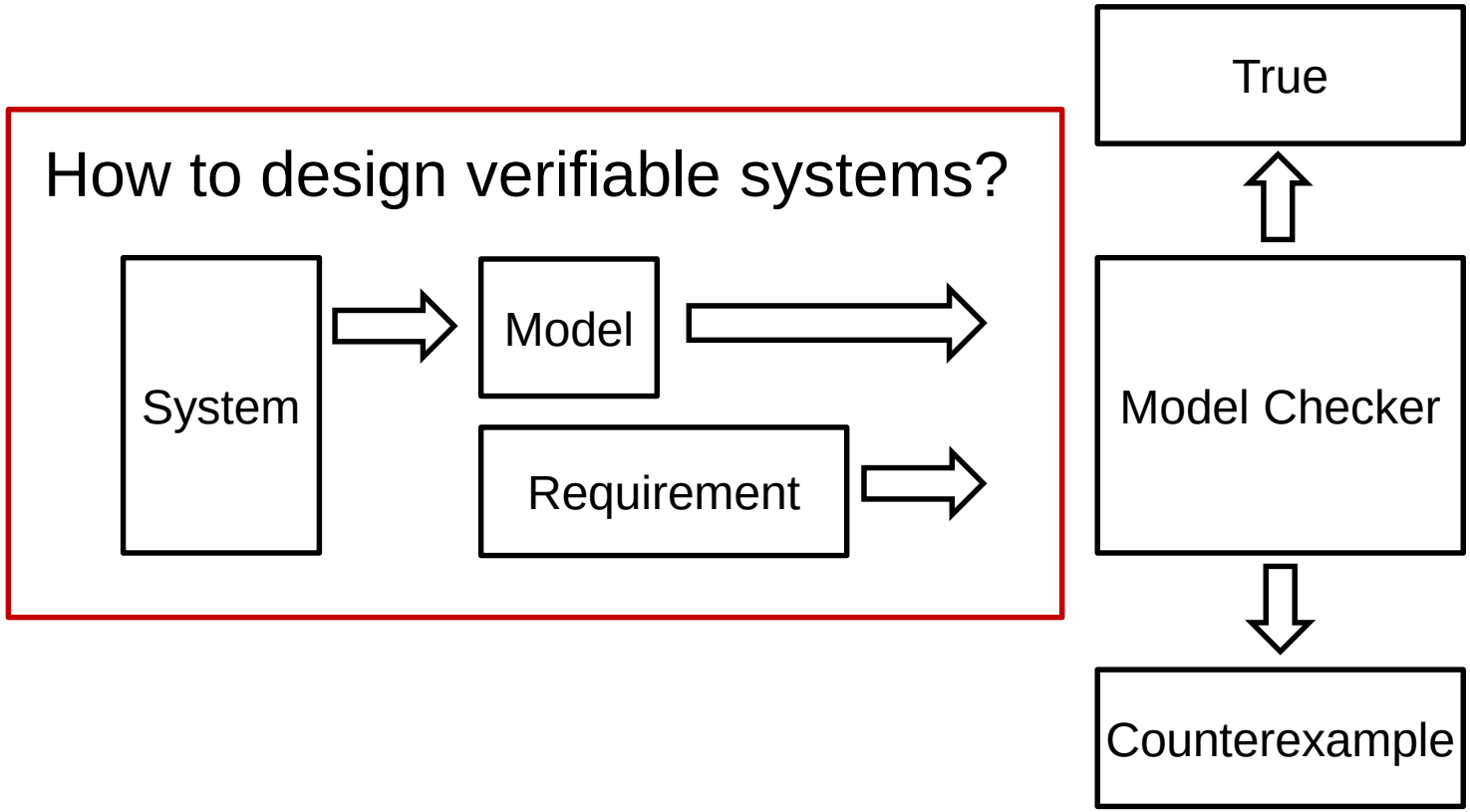
On Efficiently Specifying Models for Model Checking

Mykhaylo Nykolaychuk, Michael Lipaczewski, Tino Liebusch, Frank Ortmeier

Introduction

- **HOW TO ANALYZE A GIVEN MODEL WELL?**



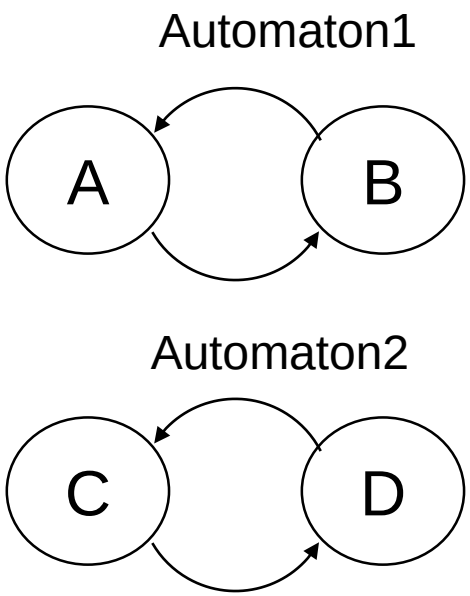
Problem Definition

- **State-spaceexplosion**
  - Exponentialincrease in memoryusageandchecking time withincreasingmodelsize
  - Composition of$n$processeswith$m$stateseachresults in m^nstates
- **Straightforwardmodelingapproach**
  - All system partsandprocessesasautomata

Simple Example

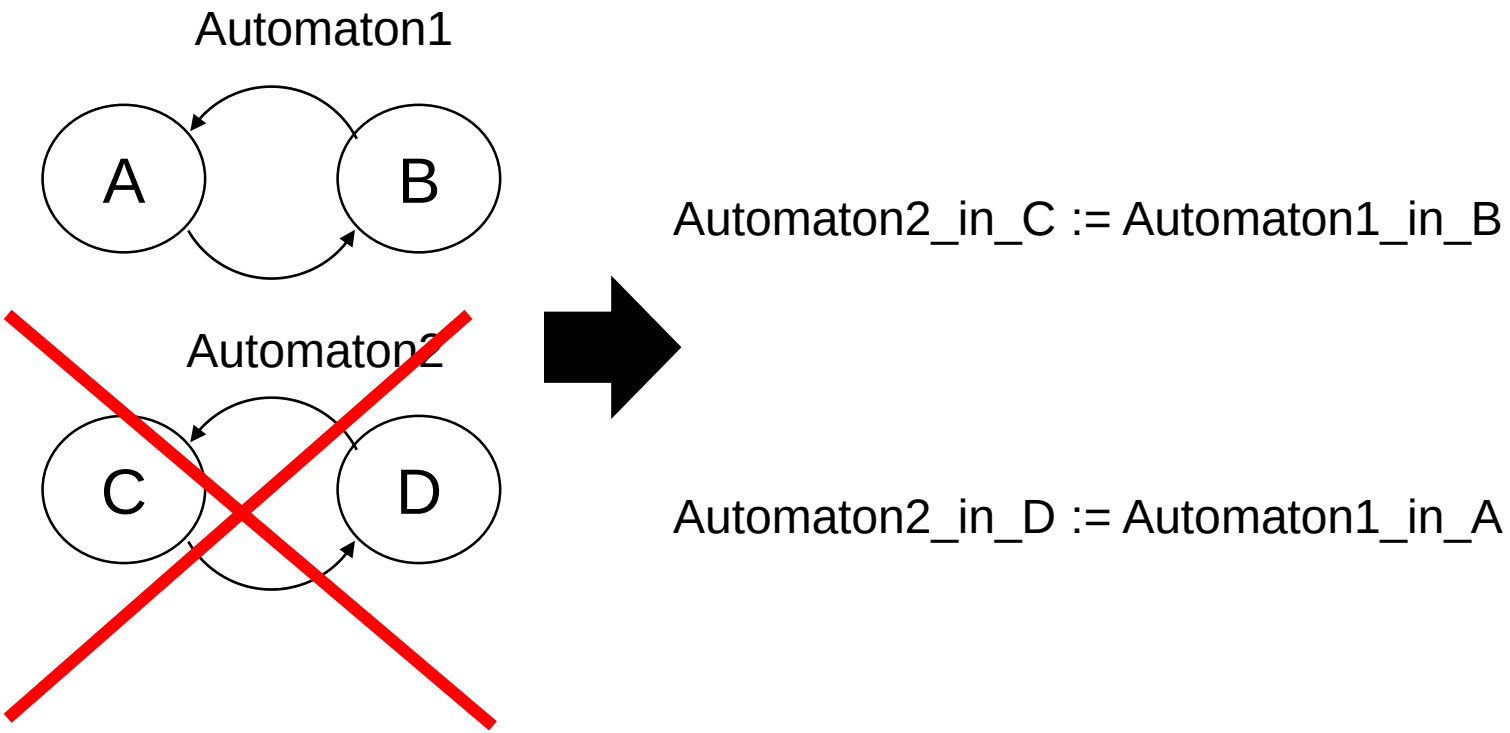
- Example:
  Two components periodically switch their states.

- **Naive modelling: 2 automata**



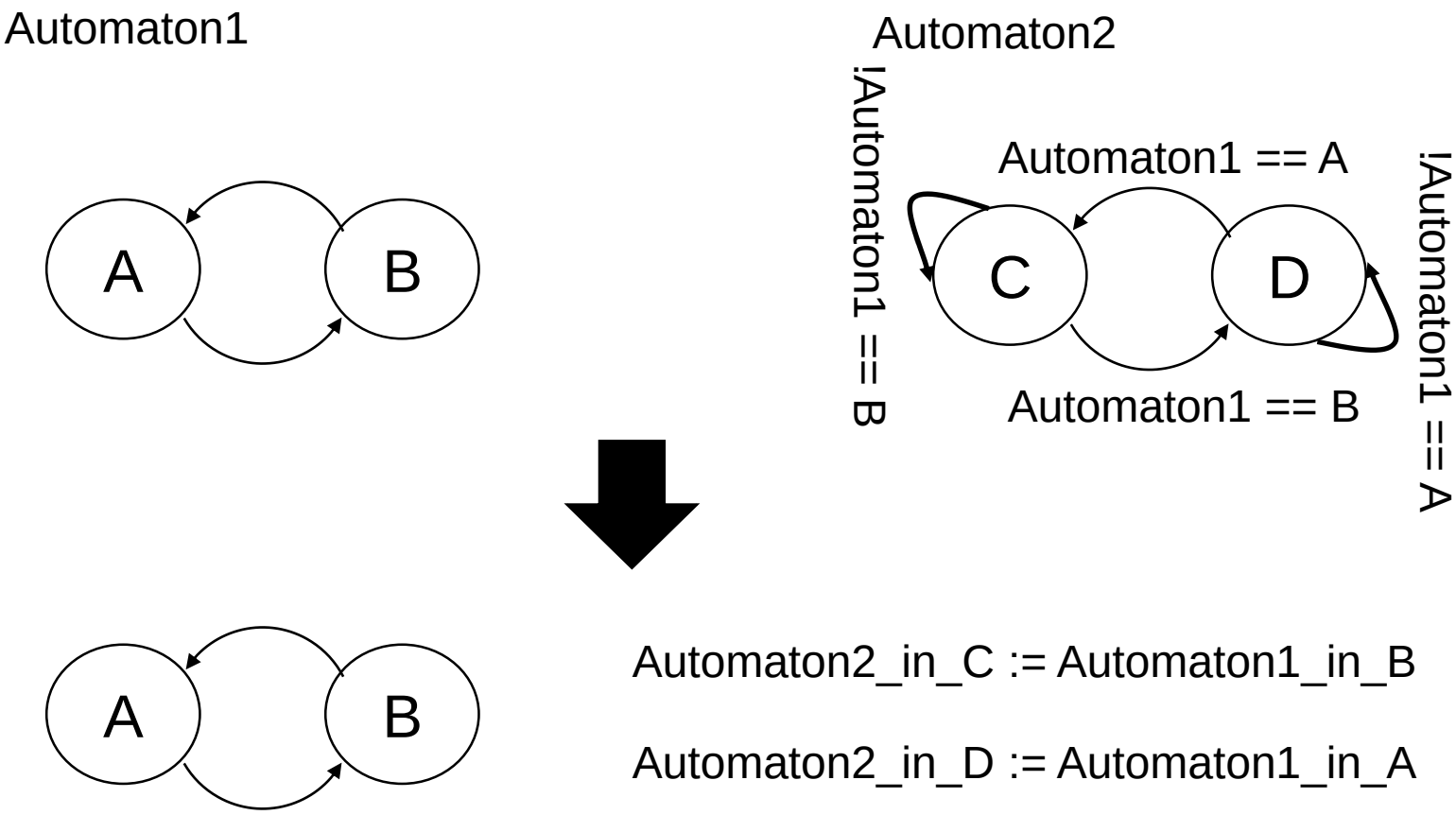
Less naive modelling

- **Replace specific automata by boolean formula during the modeling process**



Another example

- One component triggers another (simple) component



But....

- Semantical difference

Before replacement

	A1_A	A1_B	A2_C	A2_D
t1	0	1	0	0
t2	0	1	1	0

After replacement

	A1_A	A1_B	A2_C	A2_D
t1	0	1	1	0
t2	0	1	1	0

e.g., A1_A is changed its state to A1_B

PROBLEM???

What's the „right" semantics?

- Formal models are derived from informal specifications

- Question:
  Is the time delay intended/required?

  OR

  Is the time delay only optional (and might have been caused by the modelling framework)?

➔ **Guideline:** If in doubt, use formulas!

Our approach

- **Algorithm:**
  1. Use straightforward approach for initial system modeling
  2. Localize replaceable automata that only forward information
  3. Check if delay was intended and if not replace them by boolean formula and customize such substitution throughout the model

     Necessary for consistent and efficient modelling:
     - Clear namespaces
     - Formula nesting

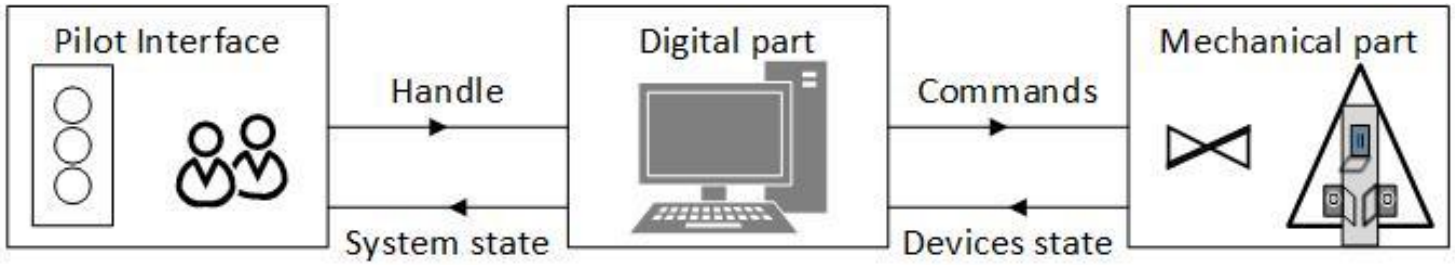
  4. Check for cyclic dependencies

SAML - VECS

- System Analysis Modeling Language (SAML)
  - Unifed qualitative and quantitative aspects of a system in a single model
- Verification Environment for Critical Systems (VECS)
  - Eclipse-based tool for design and verification of system models in SAML
  - Transormation of SAML model into input language for verification by existing tools (e.g., NuSMV, PRISM)

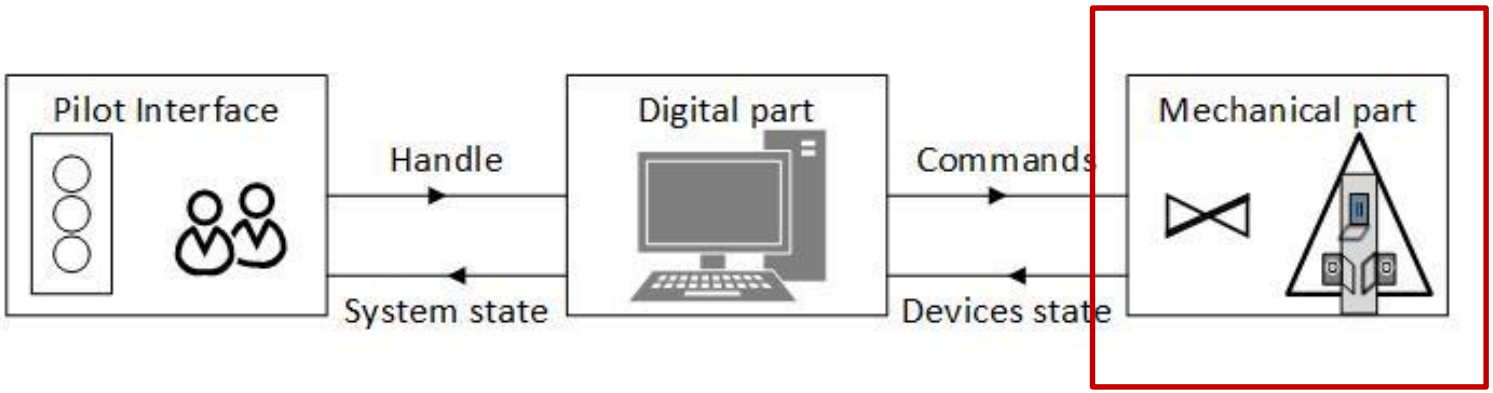
Case Study

- Landing gear system – for maneuvering the gears and associated doors
- Three main componetnts: pilot interface, digital part and mechanical part

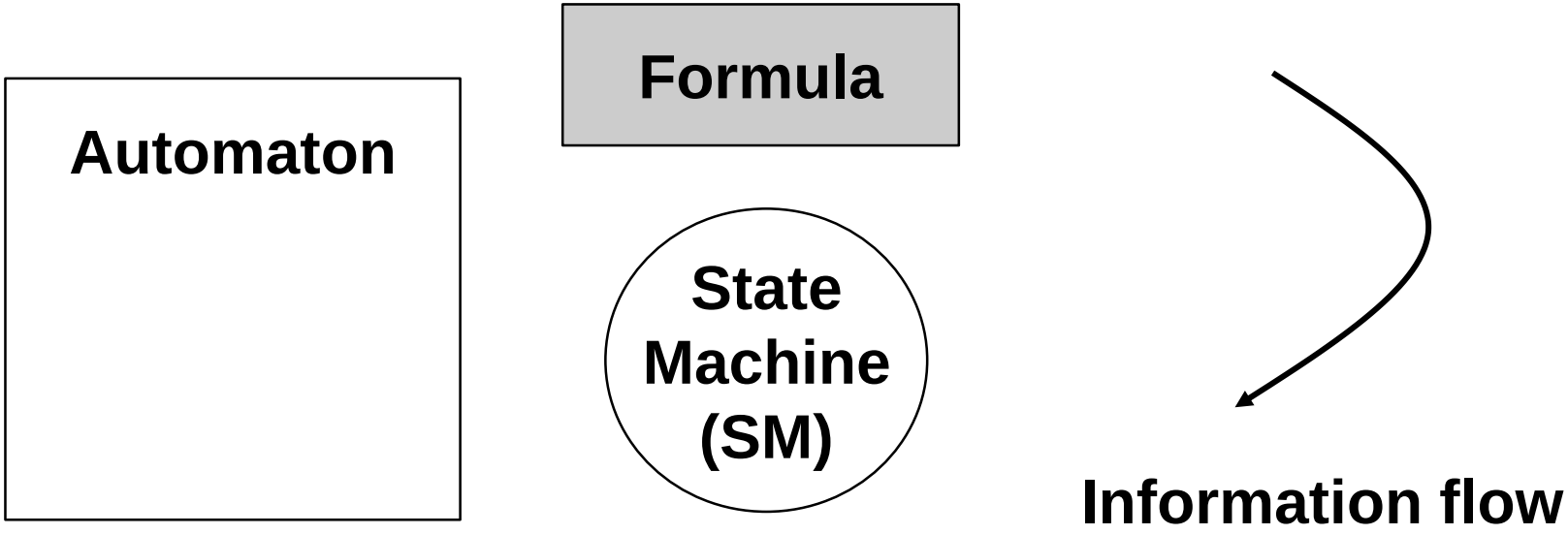


Landing Gear SAML Model

1. Standard model – all components as automata



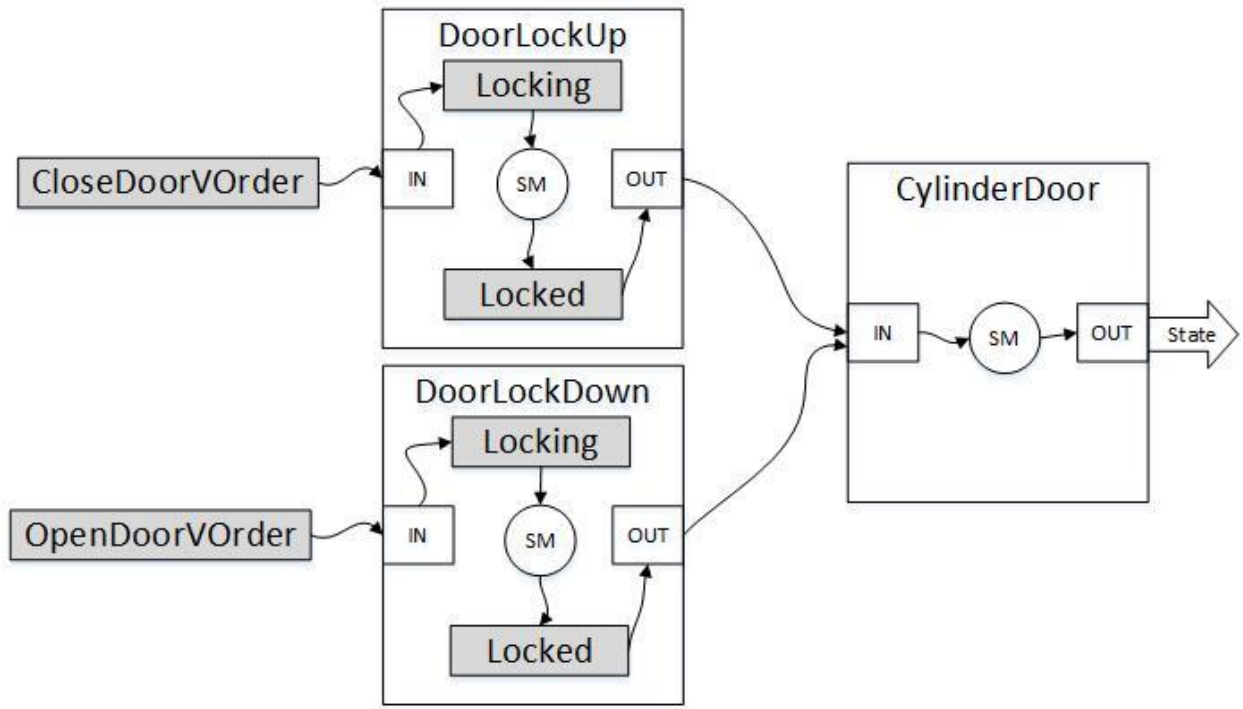
Notation




```
enum cylinder := [down , high , intermediate];

component CylinderDoor  // Automaton

                cylinderState : cylinder init cylinder.down;      // State machine
                // ... update rules for state machine

endcomponent
```

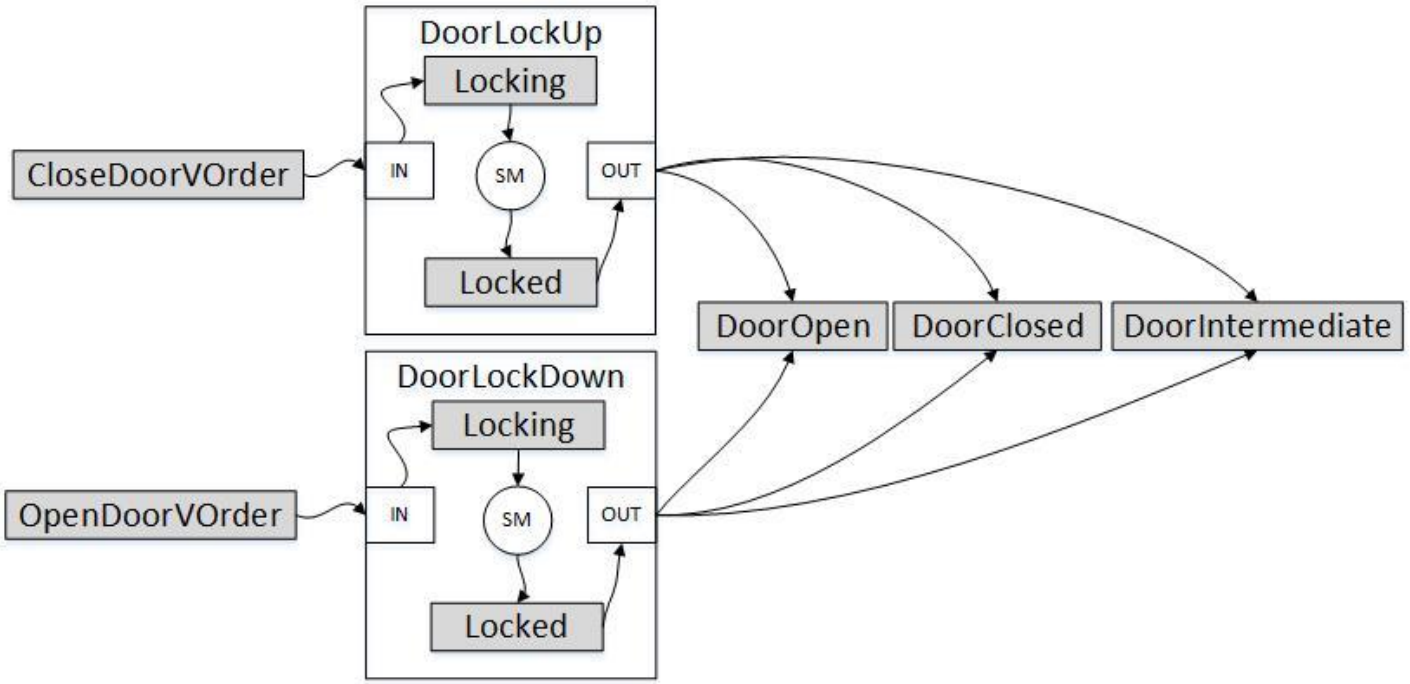
Landing Gear SAML Model

2. Localize replaceable automata
   - CylinderDoor automaton
     - States: down, high, intermediate



Landing Gear SAML Model

3. Replace chosen automaton by boolean formula



Landing Gear SAML Model

4. Check for cyclic dependencies
   - Formula may refer to itself
   - Formula could contain other formulas which may refer to this formula

Evaluation

- RequirementasCTL
- Comparisonof Standard modelandRestructuredmodelby
  - Reachablestates
  - BDD size
  - Checking time

	Reachable States	BDD nodes	Time, s
SM	2^{79}	3830081	20
RM	2^{68}	806275	0.4

- Results:
  - Reachablestatesarereducedbyfactor2^{11}
  - BDD sizeissmallerbyfactor**4.5**
  - Checking time issmallerbyfactor**45**

Conclusion and further work

Conclusion

- New directive for system design
- Semantic change
  - time delay after replacement
- Model size reduction
  - BDD and checking time become smaller

Further work

- Scalability of our approach
- Ability to find candidates for replacement automatically

Questions?