University of BRISTOL

Universität Bremen

Complete Agent-driven Model-based Testing for Autonomous Systems

Kerstin I. Eder, Wen-ling Huang, and Jan Peleska

Kerstin.Eder@bristol.ac.uk huang@uni-bremen.de peleska@uni-bremen.de

FM AS
Formal Methods for Autonomous Systems

2021-10-21

HiDyVe

Acknowledgements

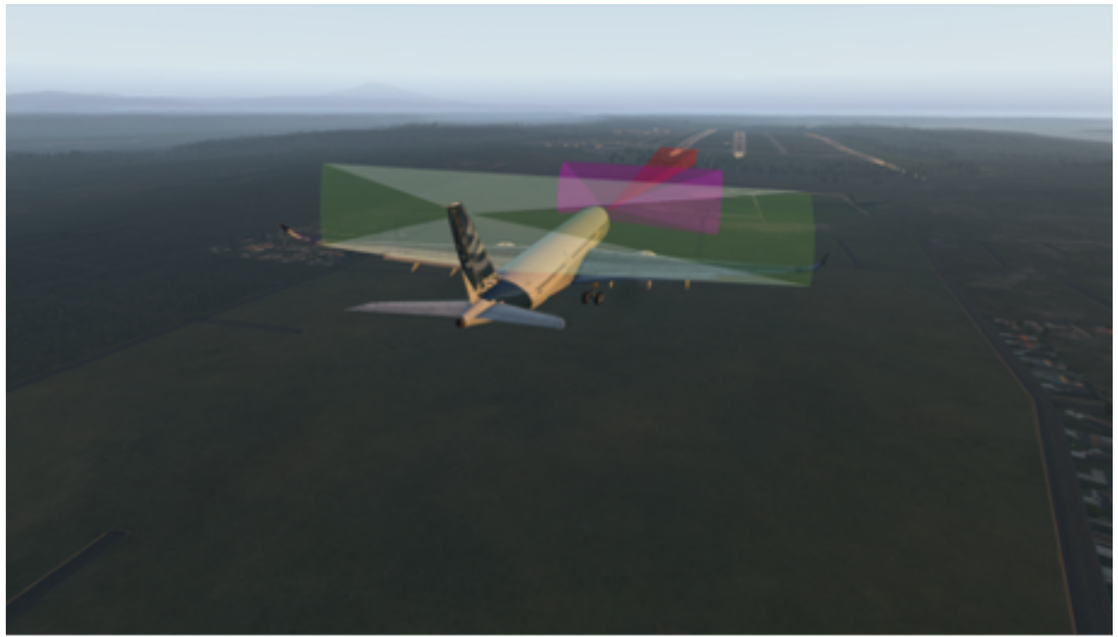
Kerstin Eder was supported in part by the “UKRI Trustworthy Autonomous Systems Node in Functionality” under grant number EP/V026518/1.

Wen-ling Huang and Jan Peleska were supported in part by the German Ministry of Economics, Project “HiDyVe – Highly Dynamic Virtual and Hybrid Validation and Verification” under grant agreement 20X1908E.

HiDyVe Project Objectives

V&V for the following application scenarios

- Formation flight – similar to platooning of trucks
- Autonomous taxi, take-off, and landing ATTOL
- Future urban mobility – combined autonomous cars and drones



Motivation &
Main Contributions

Motivation

Autonomous Transportation Systems

Why V&V for ATS is not feasible with “conventional” methods

- Too many test cases to ensure safety by means of tests on target system alone
- Applications based on machine learning (e.g. trained neural networks for image recognition) cannot be verified by means of conventional reasoning (e.g. Hoare Calculus) and may show evolving behaviour which cannot be specified “once-and-for-all”
- Multi-agent systems developing and exchanging plans on-the-fly cannot be verified and validated “once and for all” at type certification – re-validation and re-certification at runtime is required

Nidhi Kalra & Susan M. Paddock (2016): **Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?** RAND Corporation, Santa Monica, CA, doi:10.7249/RR1478.

Motivation

Why V&V for ATS is not feasible with “conventional” methods

- **Too many test cases to ensure safety by means of tests on target system alone**
- Applications ba[illegible]ined neural networks for image recogniti[illegible]of conventional reasoning (e.g. Hoare Calculus)
- Multi-agent systems developing and exchanging plans on-the-fly cannot be verified and validated “once and for all” at type certification – re-validation and re-certification at runtime is required

This is the main topic of this talk – and we suggest a comprehensive approach to solve this problem

Nidhi Kalra & Susan M. Paddock (2016): **Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?** RAND Corporation, Santa Monica, CA, doi:10.7249/RR1478.

Note. This is a position paper – feedback is very welcome!

A new Strategy to Perform ATS Testing

An approach to solve the test suite size problem for ATS

- On the module level, use **complete model-based testing strategies** with guaranteed fault coverage
- On the system level, use novel **scenario-based end-to-end testing strategy** and novel **strategy to assess system test coverage**, exploiting knowledge about complete module tests and their models
- Optimise the system test execution by
  - Multiple concurrent system **test executions on target systems and in the cloud**
  - Change of system test case objectives on-the-fly (**online testing**), driven by **continuous coverage assessment**
  - Coordination of system test executions by means of multi-agent system (**agent-based system testing**)

This Approach is Suitable to Obtain Certification Credit

We analyse existing standards and try to predict future changes

- Automotive domain
  - ISO 26262. Road vehicles – functional safety
  - ISO 21448. Road vehicles – Safety of the intended functionality
- Railway domain
  - CENELEC EN 50128:2011. Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems

This Approach is Suitable to Obtain Certification Credit

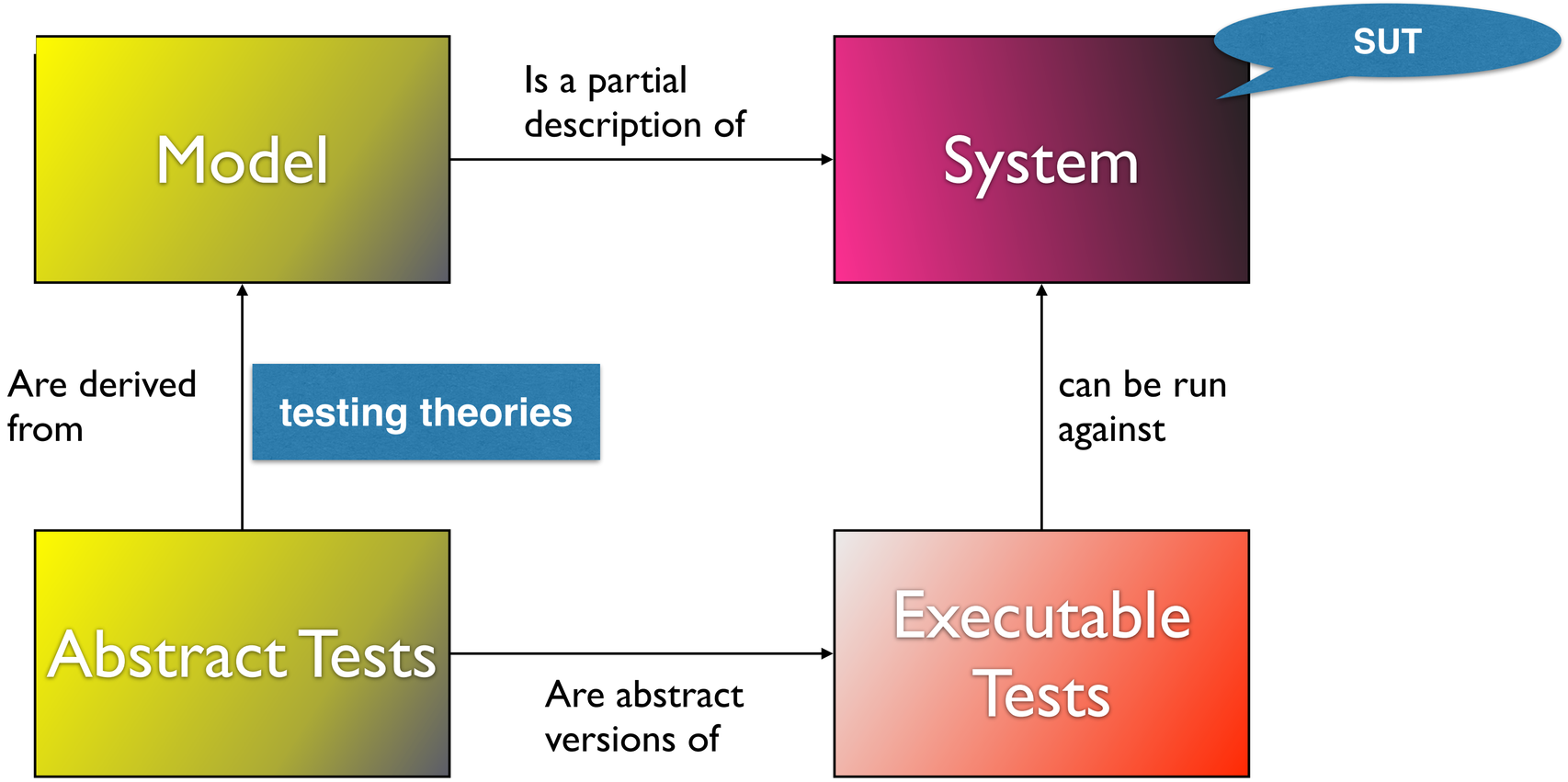
We analyse existing standards and predict future changes

- Avionic domain
  - RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification
    - DO-330. Software Tool Qualification Considerations
    - DO-331. Model-based Development and Verification Supplement
    - DO-333. Formal Methods Supplement
  - EASA (European Union Aviation Safety Agency)
    - Artificial Intelligence Roadmap – A human-centric approach to AI in aviation
    - Concept Paper: First usable guidance for Level 1 machine learning applications

Complete Model-based Testing Methods

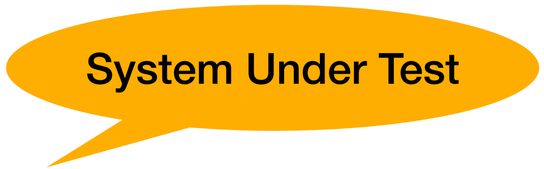
Model-Based -Testing

MBT-Paradigm



Conformance vs. Property-oriented Testing

In the context of safety-critical control systems



- **Conformance testing**
  - **Objective**. Verify that SUT behaviour conforms to that of a reference model
  - **Conformance relations**
    - Observational equivalence
    - Refinement in several variants
  - **Modelling formalisms**
    - Finite state machines and variants thereof
    - Process algebras
    - UML/SysML
    - . . .
  - **Test cases** are derived from the model

Conformance vs. Property-oriented Testing

In the context of safety-critical control systems

- **Property-oriented testing**
  - **Objective**. Verify that SUT fulfils a behavioural property
  - **Property specification**
    - Temporal logic (LTL, CTL, TCTL, …)
    - Test cases are **derived from the property**
  - **Model-based property-oriented testing**
    - In addition to the property specification, a model specifies the complete required behaviour of the SUT or a part thereof
    - The model fulfils the property
    - Test cases are **derived from the property in combination with the model**
    - The existence of a model helps to **reduce the test suite size**

Complete Testing Theory

in conformance testing

- A **testing theory** is a recipe explaining how to derive test cases from a model, so that the detection of conformance violations by the SUT can be guaranteed under certain hypotheses
- A test suite (generated by application of the theory) is **exhaustive**, if every non-conforming SUT fails at least one test case of the suite
- A test suite is **sound**, if every conforming SUT passes every test case
- A test suite is **complete**, if it is sound and exhaustive

Complete Testing Theory

in property-oriented model-based testing

- A **testing theory** is a recipe explaining how to derive test cases from a model and a property formula, so the detection of property violations by the SUT can be guaranteed under certain hypotheses
- A test suite (generated by application of the theory) is **exhaustive**, if every SUT violating the property fails at least one test case of the suite
- A test suite is **sound**, if every SUT fulfilling the property passes every test case
- A test suite is **complete**, if it is sound and exhaustive

Conformance vs. Property-oriented Testing

Academia vs. Industry

- In academia, conformance testing has been investigated comprehensively
- **In industry, the focus is on model-based property-oriented testing**
  - The ATS-related standards demand that every test case is traced back to a requirement (= property)
  - Models are created anyway for the purpose of system design
  - The concept of conformance does not occur anywhere in the ATS-related standards
  - Complete real-world systems are too complex to investigate model conformance
  - RTCA DO-178C:
    “Model coverage analysis does not eliminate the need for traceability analysis between requirements from which the model was developed and the model.”

Testing on the Module Level

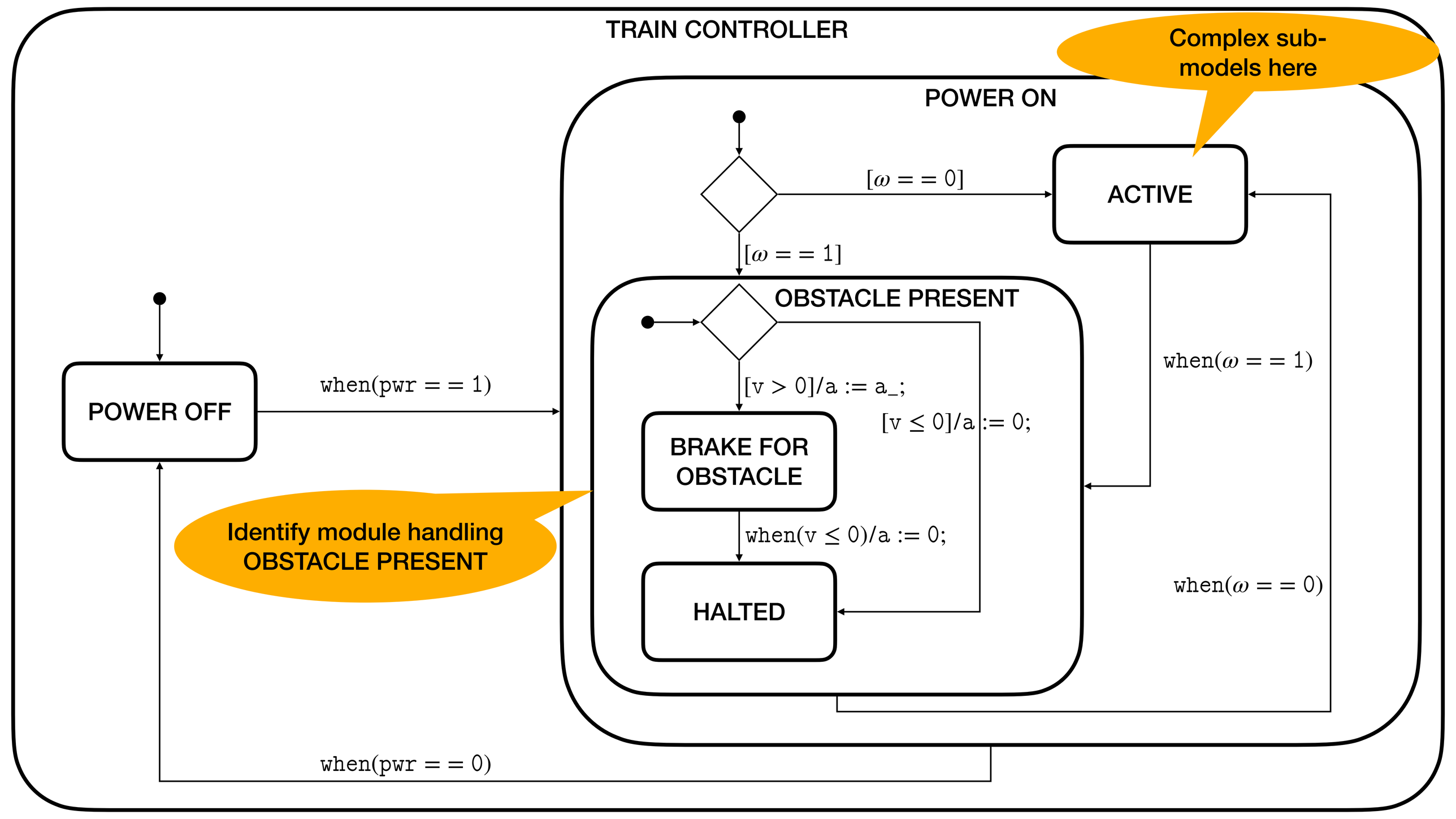
Applicability of Complete Testing Methods

on the module level

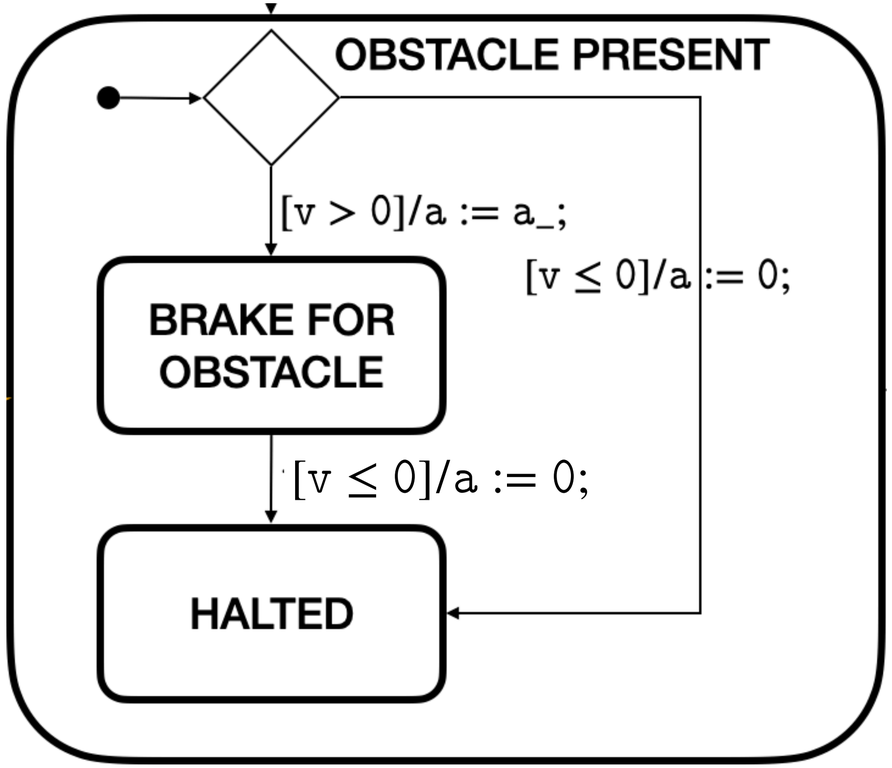
- On the system level, behavioural models are usually too large and complex to allow for complete testing methods
- On the module level, the model size is acceptable
- **Example**. Consider system of autonomous freight train controller

Detailed system model available in

Kerstin Eder, Wen-ling Huang & Jan Peleska (2021): **Complete Agent-driven Model-based System Testing for Autonomous Systems** – Technical Report, doi:10.5281/zenodo.5203111.


TRAIN CONTROLLER
Complex sub-models here
POWER ON
ACTIVE
[ω == 0]
[ω == 1]
OBSTACLE PRESENT
when(ω == 1)
POWER OFF
when(pwr == 1)
[v > 0]/a := a_;
[v ≤ 0]/a := 0;
BRAKE FOR OBSTACLE
when(v ≤ 0)/a := 0;
HALTED
when(ω == 0)
Identify module handling OBSTACLE PRESENT
when(pwr == 0)

Reference model for module ObstaclePresent (Symbolic Finite State Machine [SFSM])



SFSM

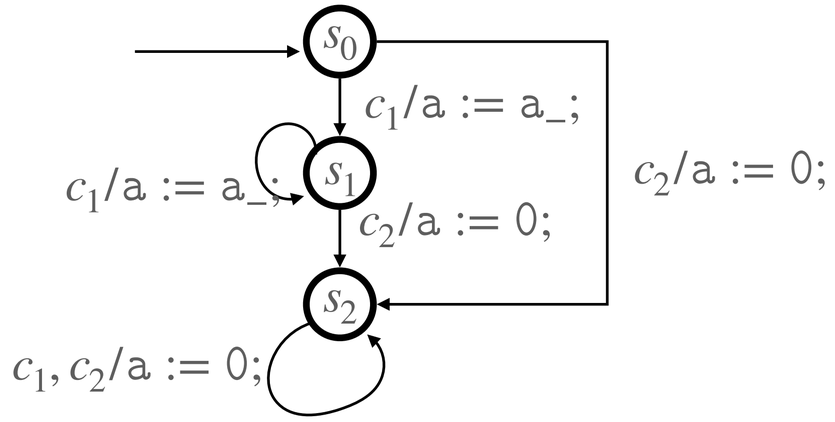
Objective. Verify that software is I/O-equivalent to SFSM Model

Application of complete testing strategy for conformance testing

1. Create input equivalence classes from guard conditions [check module code whether other/finer guards are used and refine classes if necessary]

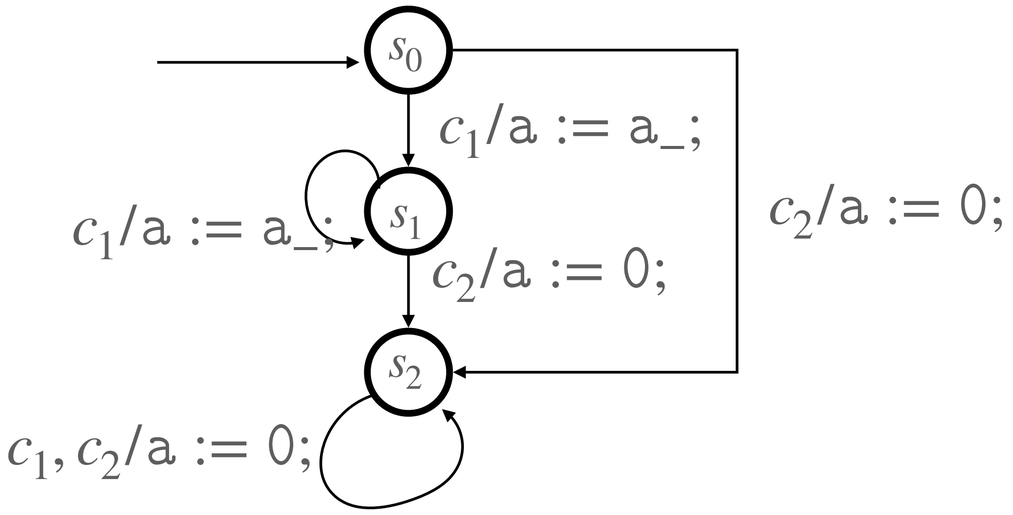
$$\mathcal{I} = \{c_1, c_2\},\ c_1 = \{\mathrm{v} > 0\},\ c_2 = \{\mathrm{v} \leq 0\}$$

2. Create abstraction SFSM → FSM



Wen-ling Huang & Jan Peleska (2016): **Complete model-based equivalence class testing.** Software Tools for Technology Transfer 18(3), pp. 265–283, doi:10.1007/s10009-014-0356-8.

FSM abstraction of module ObstaclePresent



Theorem. Test case translation from FSM to SFSM preserves completeness

Wen-ling Huang & Jan Peleska (2016): **Complete model-based equivalence class testing.** Software Tools for Technology Transfer 18(3), pp. 265–283, doi:10.1007/s10009-014-0356-8.

Wen-ling Huang & Jan Peleska (2017): **Complete model-based equivalence class testing for nondeterministic systems.** Formal Aspects of Computing 29(2), pp. 335–364, doi:10.1007/s00165-0160402-2.

Application of complete testing strategy for conformance testing

4. Perform static analysis of module code regarding bound m for number of control states

5. Calculate complete test suite from FSM, observing m

   TC 1: $c_1 \,.\, c_1$

   TC 2: $c_2 \,.\, c_1 \,.\, c_1$

   TC 3: $c_2 \,.\, c_2 \,.\, c_1$

   Calculated for $m = 0$

   Observe that s_0 and s_1 are equivalent!

6. Calculate concrete test suite by selecting 1 representative per input class

   TC 1: $(v = 10) \,.\, (v = 10)$

   TC 2: $(v = 0) \,.\, (v = 10) \,.\, (v = 10)$

   TC 3: $(v = 0) \,.\, (v = 0) \,.\, (v = 10)$

7. Expected results are checked against reference SFSM

Property-oriented Module Testing

Skipped in this Presentation

- See paper and technical report doi:10.5281/zenodo.5203111
- The properties (= requirements) we would like to prove for the ObstraclePresent module are

$$\mathbf{G}(v > 0 \Rightarrow \mathbf{X}a = a\_)$$

$$\mathbf{G}(v = 0 \Rightarrow \mathbf{X}a = 0)$$

System-Level Testing

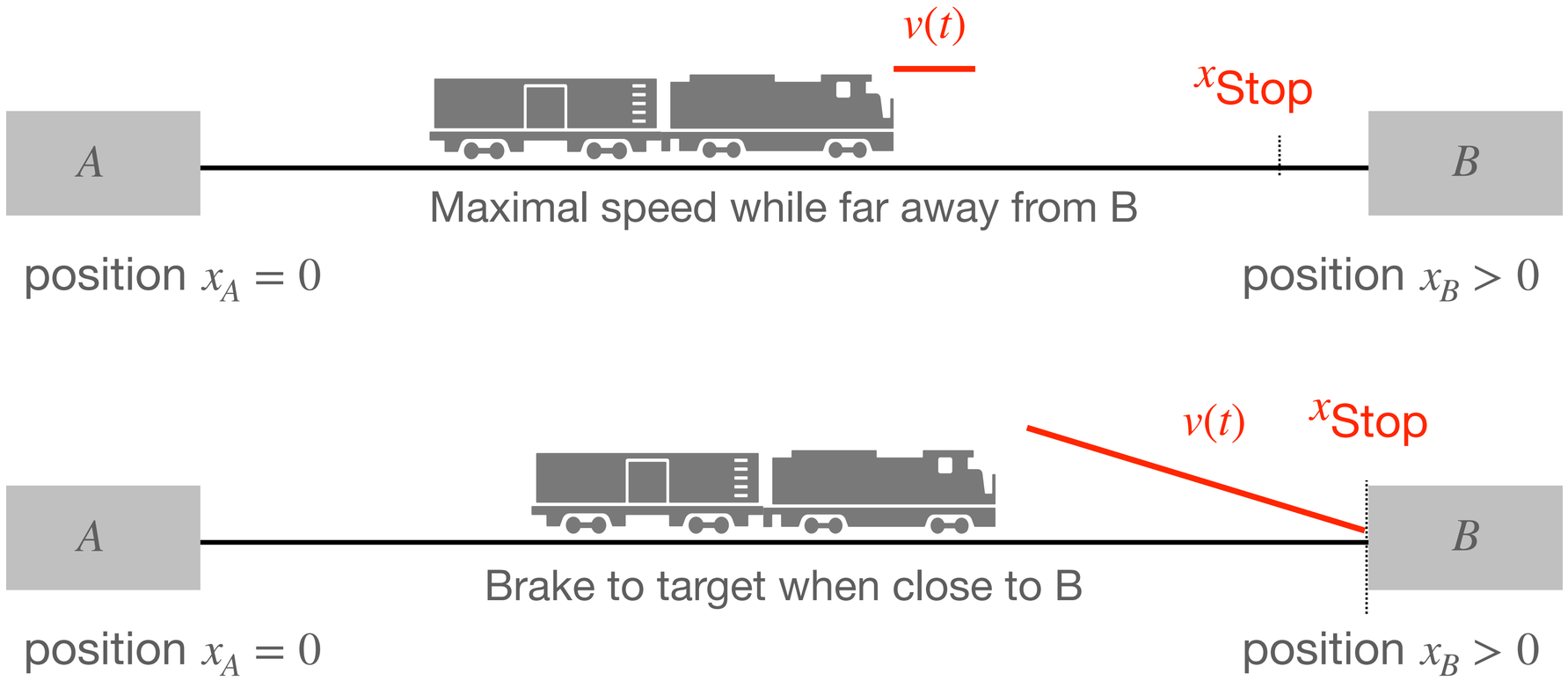
System-Level Testing

Meaningful System Tests

- On the system-level, **end-to-end (E2E) tests**, describing meaningful scenarios from the end users' perspective, are required → they need to be suitable for **validation**
- Significant observation: **complete module tests are not suitable for E2E test construction**, because the paths selected for module tests are optimised with respect to length
  - These shortest paths often specify robustness situations
  - Module tests require frequent resets, due to this shortest path selection, they are not designed for in-depth investigation of paths
  - SFSMs used as module test reference models do not contain sufficient information to construct meaningful E2E tests

System-Level Testing

Meaningful E2E Tests

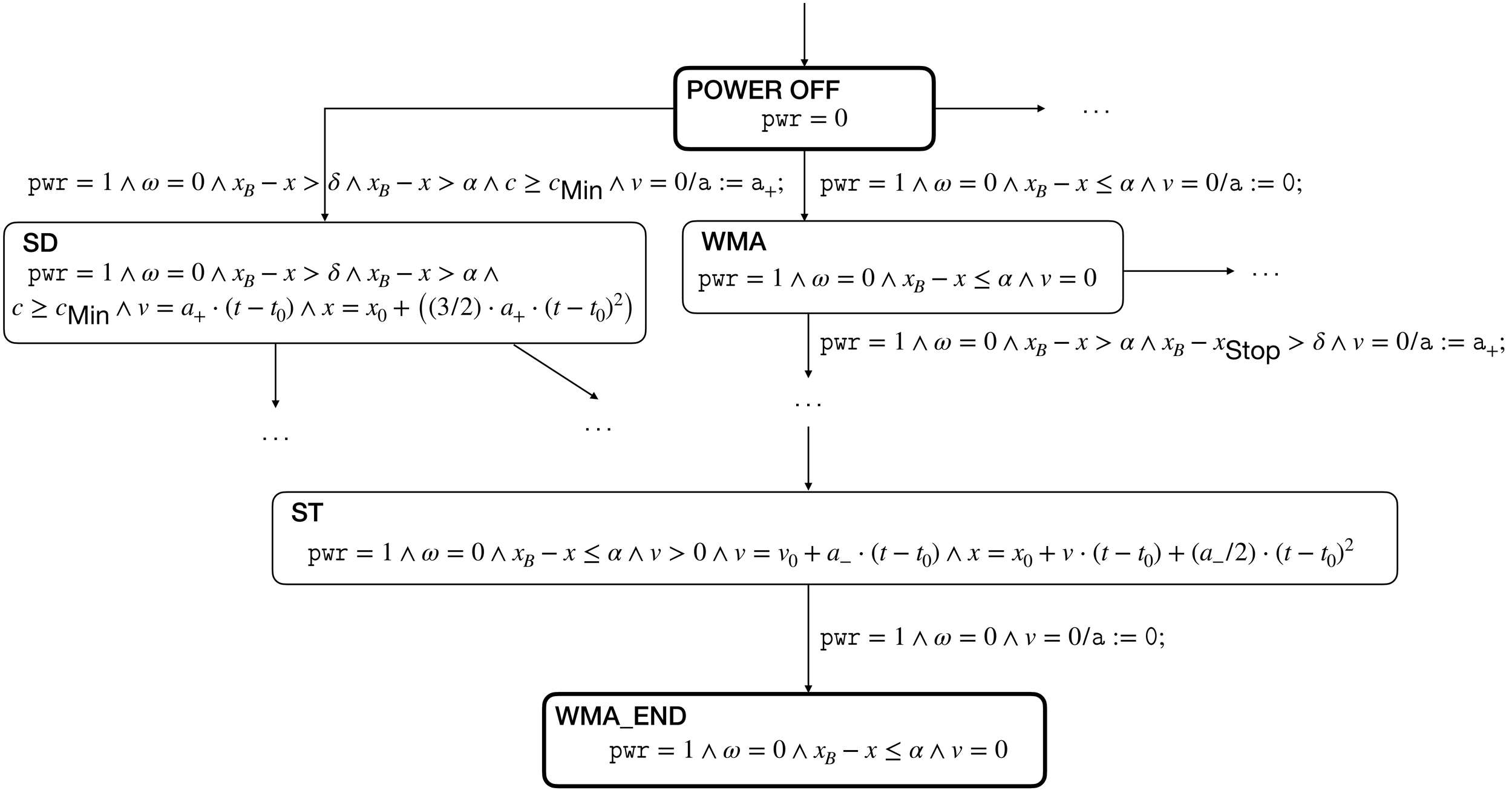


System-Level Testing

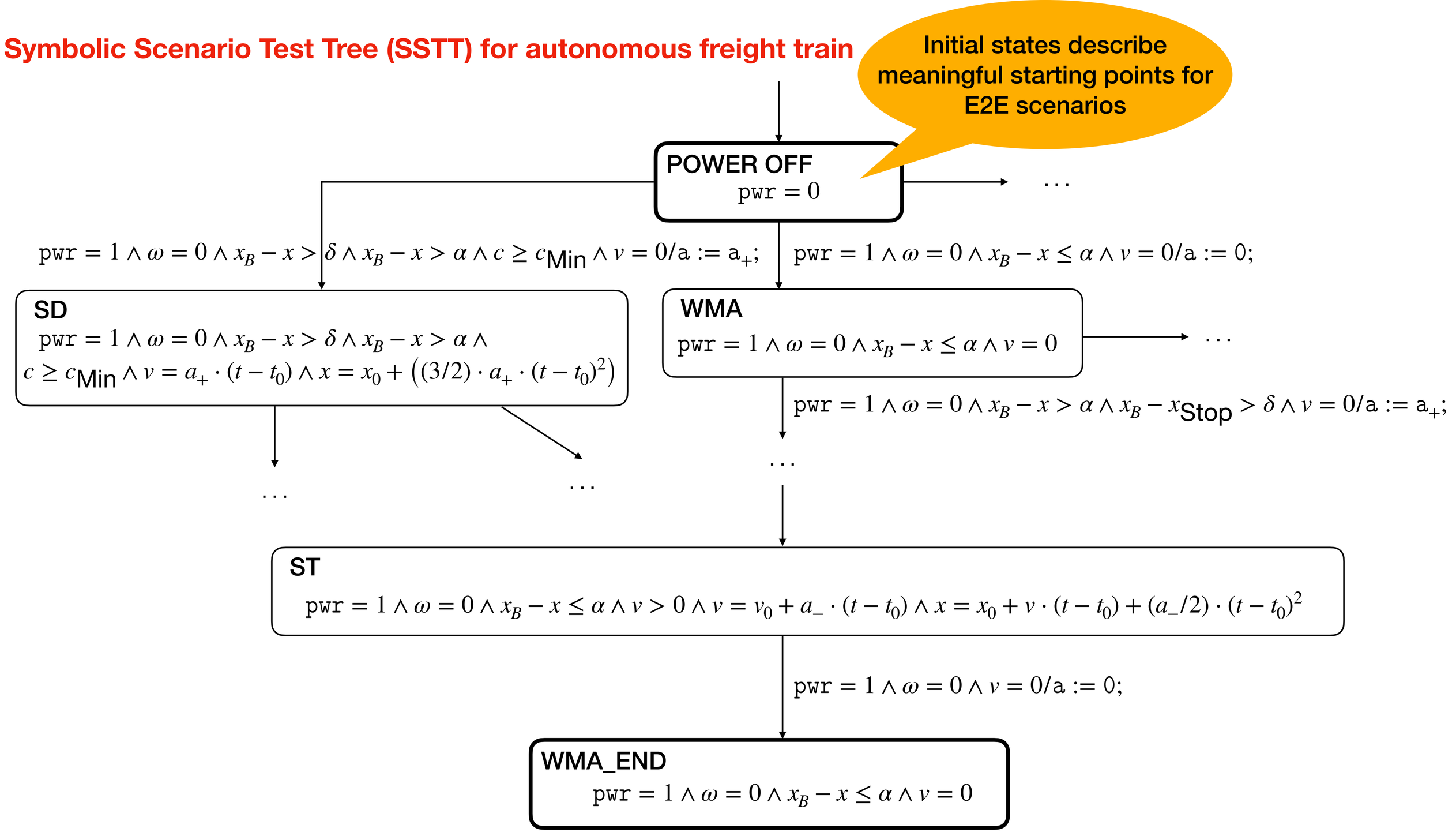
E2E Test Scenarios

- Use **Hybrid Automata (HA)** instead of SFSMs
  - **HA model terminates**: it describes E2E scenarios on the **system level**
  - States are decorated by **invariants specifying time-continuous evolution of input variables** according to applicable physical laws
  - Initial states describe meaningful **starting points for E2E scenarios**
  - Termination states describe meaningful **scenario end states**
  - Transitions best tested on module level are **removed from the HA model**
  - E2E tests are derived from HA by selecting specific paths from start to end states
    - This results in a tree structure: **Symbolic Scenario Test Tree (STTT)**

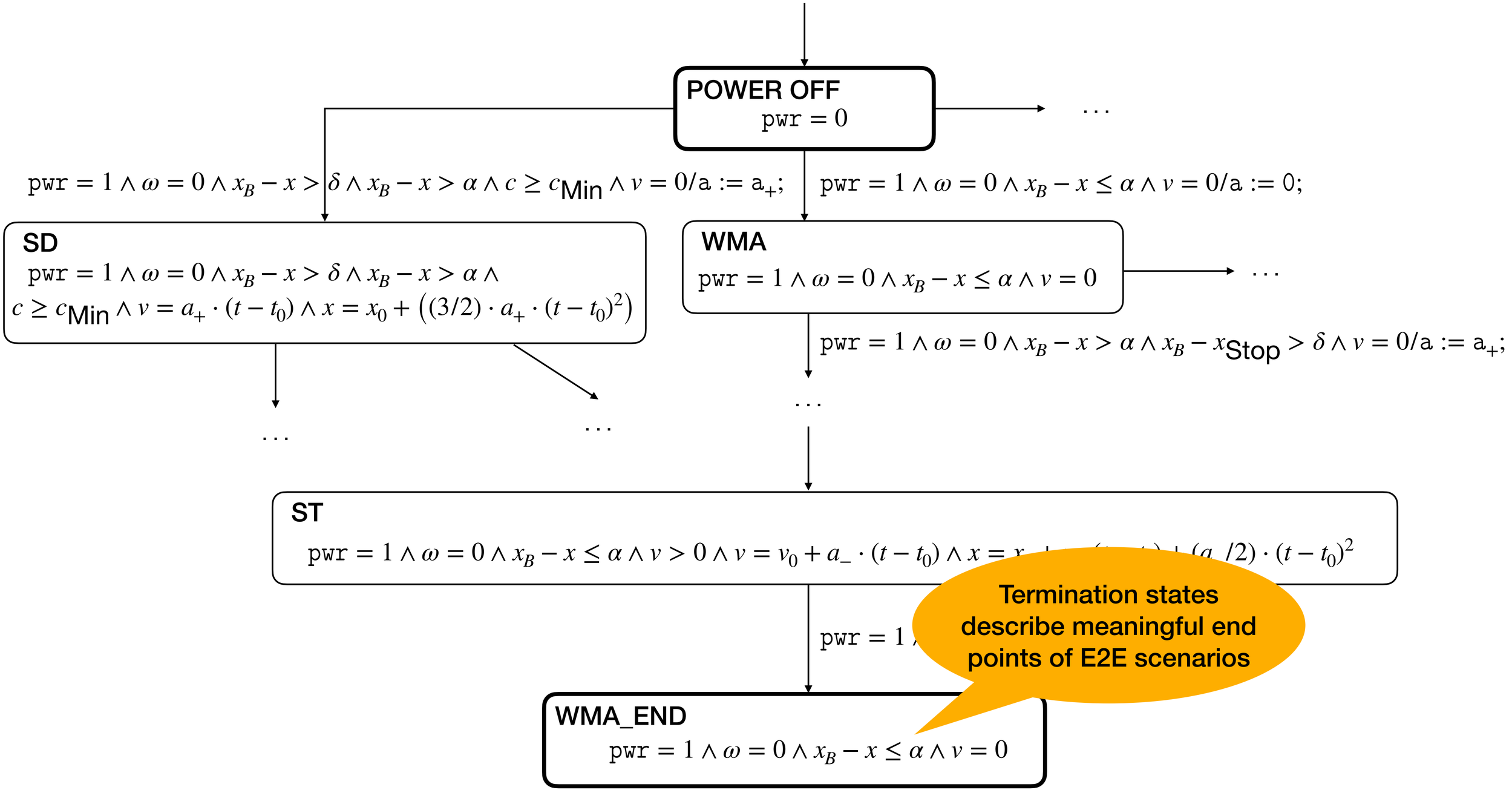
Symbolic Scenario Test Tree (SSTT) for autonomous freight train



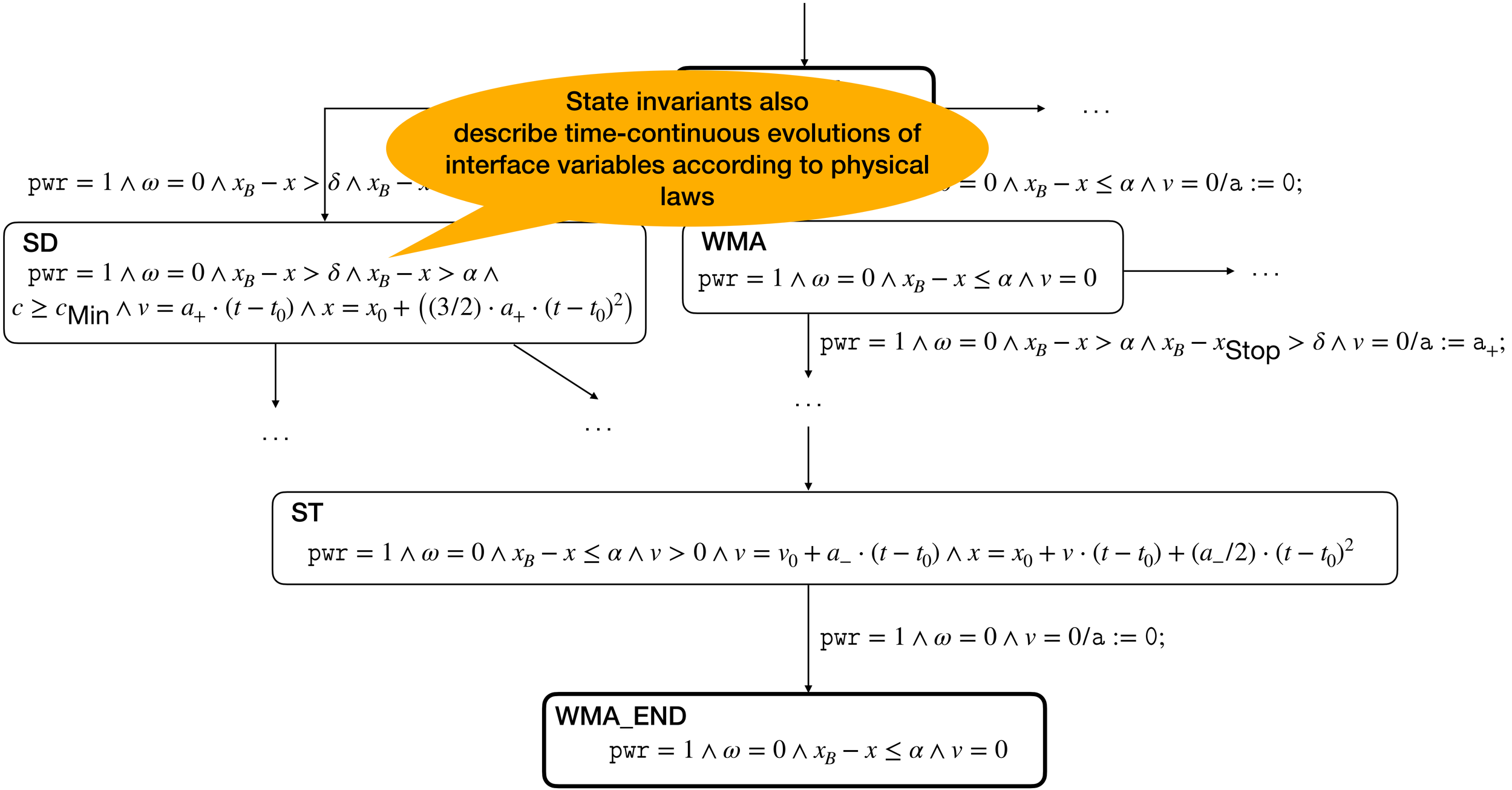
Symbolic Scenario Test Tree (SSTT) for autonomous freight train



Symbolic Scenario Test Tree (SSTT) for autonomous freight train



Symbolic Scenario Test Tree (SSTT) for autonomous freight train



Test Execution on the System Level

Test Execution of the System Level

Testing in the cloud

- Majority of tests have to be executed in the cloud, to ensure timely completion of test campaigns
- Prerequisites to obtain certification credit for test results obtained in the cloud
  - Trustworthy simulation of the “real” operational environment
  - Execution of the SUT software in trustworthy simulator (virtual prototype) modelling the target hardware (registers, address maps, …)

Vladimir Herdt, Daniel Große, Pascal Pieper & Rolf Drechsler (2020): **RISC-V based virtual prototype: An extensible and configurable platform for the system-level.** J. Syst. Archit. 109, p. 101756, doi:10.1016/j.sysarc.2020.101756.

Test Execution of the System Level

Testing on the target in the real operational environment

- System tests need to be **intrusive**, so that interfaces can be manipulated to make rare events occur more often
  - **Example**. For the autonomous freight train, the position sensor interfaces need to be manipulated, so that low-confidence position values and erroneous position values can be created whenever needed during the test execution

Test Execution of the System Level

Coverage considerations

- How many system tests need to be executed ?
  - Avionic standard RTCA DO-333 states:
    "Tests executed in target hardware are always required to ensure that the software in the target computer will satisfy the high-level requirements . . . "
- For requirements φ_i represented in LTL, suitable witness paths π_i can be identified in the SSTT (or the SSTT needs to be extended)
- System tests need to cover each of these paths π_i with at least one witness

Test Execution of the System Level

Coverage considerations

- How many system tests need to be executed **on the target system** in the real operational environment?
- For most critical applications, ensure that system tests on the target **cover all normal behaviour transitions of the module test models**
  - **Justification**. Software has already been shown to be correct with respect to module test models. Therefore, covering these model branches in system tests on target indicates that the HW/SW integration satisfies the requirement
- For less critical applications, a smaller percentage of the module test transitions should be covered by the system tests on target

Test Execution of the System Level

System test coordination by multi-agent system

- System tests are executed concurrently in the cloud and on the target in the real operational environment
- **Coordinator agent** has view on overall coverage achieved so far and on test execution status of all running tests
- Coordinator agent directs **test execution agents** to continue their on-the-fly tests into branches of the SSTT that increase the coverage in an optimised way
- In the cloud, **simulation agents** provide obstacles and position values that are consistent with the applicable physical laws

Greg Chance, Abanoub Ghobrial, Séverin Lemaignan, Tony Pipe & Kerstin Eder (2020): **An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification.** In: IEEE International Conference On Artificial Intelligence Testing, AITest 2020, Oxford, UK, August 3-6, 2020, IEEE, pp. 31–38, doi:10.1109/AITEST49225.2020.00012.

Conclusion

Conclusion

Summary

- We have proposed a novel V&V approach for complex autonomous systems
  - Complete tests on the module level ensure logical software correctness
  - Scenario tests on the system level represent meaningful end-to-end tests
  - Some scenario tests are executed with original equipment in real operational environment, the majority of tests in the cloud
  - Coverage monitoring exploits the fact that the module software has been proven to be correct
  - System tests are executed concurrently on-the-fly, coordinated by test agents

Conclusion

Next steps

- Refine the coverage theory for system testing
- Several **proofs of concept** for the V&V steps presented here
  - Autonomous freight train modelled in this paper
  - Safety-critical control of human-robot interaction
    - Please visit FMAS presentation of Mario Gleirscher: tomorrow at 13:30
  - Autonomous cars
  - Autonomous drones
- All experiments conducted with cloud tests in simulation environment and on target system hardware in realistic physical environment

THANK YOU VERY MUCH FOR YOUR ATTENTION!